第８章　SPACE のバージョンアップに対応しての変更法

8.1 はじめに

分散並列型動的解析システムは、SPACE の動的解析システムをベースにして開発されている。従って、SPACE の動的ソルバーを変更したり、バージョンアップしたりする場合、可能な限り自動的に分散並列型動的解析システムも変更されるように開発環境を整えている。しかし、システムの違いにより、手動で変更をしなければならない部分が、どうしても残ってしまう。ここでは、SPACE システムに変更があった場合、常に両システムが適合するために、どのサブルーチンを変更しなければならないかについて説明する。

8.2 分散並列型動的解析システムの開発環境

本節では、分散並列型動的解析システムの開発環境について説明する。最初に、開発環境の元になるフォルダーの構成を図 8-1 に示す。

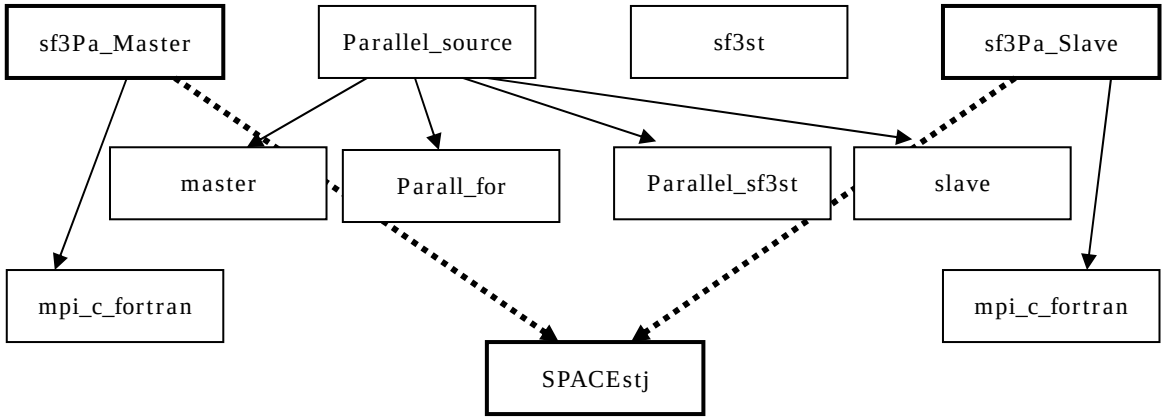


図 8-1 分散並列型動的解析システムの開発環境

図中の実線で示される矢印は、矢印方向のフォルダーが元のフォルダーに含まれることを意味する。同様に点線は、矢印方向のフォルダーに実行形式のファイルが作成されることを意味する。つまり、コンパイル・リンクすることでマスター、スレーブ共に実行モジュールがフォルダーSPACEstj に作成され、自動的に SPACE に組み込まれることになる。

マスター用動的解析システムのワークスペースやその他のサブルーチンをコンパイル・リンクするための情報は、全て sf3Pa_Master フォルダーに保持されている。また、このフォルダーには、C++で書かれた動的解析システム内のグラフ処理やウインドウ管理などを行う関数が

保存されている。同じく、スレーブ用の動的解析システムのワークスペースやその他のサブルーチンをコンパイル・リンクするための情報は、全て sf3Pa_Slave フォルダーに保持されている。また、このフォルダーには、C++で書かれた動的解析システムが保存されている。

次に、各フォルダー内にどのようなサブルーチンや関数が保存されているかについて説明しよう。各フォルダーの内容が表 8-1 にまとめられている。

表 8-1　フォルダーの内容

フォルダー名	解説	適用
sf3st	SPACE の動的解析システム全体が保存されており、必要部分が開発用プロジェクトに登録されている。従って、ここで登録されている部分については、変更は自動的に行われる。	自動的に変更されているので、ここでは特に変更する必要なし
sf3Pa_Master	マスター用動的解析システムを開発するための全ての情報が保存されている。C++で書かれた図形処理用などの関数が存在する。	変更の必要なし
sf3Pa_Slave	スレーブ用動的解析システムを開発するための全ての情報が保存されている。C++で書かれた関数が存在する。	変更の必要なし
mpi_c_fortran	MPI_c ライブラリを使用する場合、Fortran 用のインターフェイスとインクルードファイルがある。	変更の必要なし
Parallel_source	４つのフォルダーが存在する。	変更の必要なし
master	マスター用動的ソルバーである submain_pa.f ファイルがひとつ存在し、そのファイルには **submain_dynamic_a** サブルーチンが収められている。	**動的ソルバーの主サブルーチンが変更になった場合は、ここを変更する。**
Parall_for	送受信用のサブルーチンなどが収められており、並列処理専用のサブルーチンである。データ送受信するためにデータをパックするコードを加えたサブルーチンがある。	**データ送受信するためのデータパックを行ったサブルーチンは、変更する。**
Parallel_sf3st	SPACE の動的解析システムのサブルーチンを並列用に変更を加えたサブルーチンが収められている。	**sf3st のサブルーチンが変更になった場合、該当するサブルーチンを変更する。**
slave	スレーブ用動的ソルバーである submain_pa.f ファイルがひとつ存在し、そのファイルには **submain_dynamic_S** サブルーチンが収められている。	**動的ソルバーの主サブルーチンが変更になった場合は、ここを変更する。**
SPACEstj	SPACE システムの実行形式が保存されており、マスター、スレーブ用動的解析システムが、ここに作成される。	

表 8-1 で分かるように、SPACE の動的解析システムが変更になった場合、次に示す 4 つのフォルダー内のサブルーチンを変更しなければならない。

1．master
2．slave
3．Parall_for
4．Parallel_sf3st

8.3 変更すべきサブルーチン

本節では、変更すべきサブルーチンについて説明する。変更用サブルーチンは、全て元の SPACE 動的ソルバー用に手を加えて変更したものである。サブルーチン名も元の名前の末尾に「_pa」を付けているため、元の名前も簡単に分かるようになっている。

この変更すべきサブルーチンは、いくつかに分類できる。ひとつは、並列処理用に変更した部分が少なく、多くは、次のような部材の分担部分である。

SPACE 動的解析システム

```
      subroutine Add_damp2_ld(nx,ld_point,Member,n_member,
     *       past_disp_point,past_vel_point,past_acc_point,
     *       am_member,Newmark_P,Element,load_mass)
       .
      do i=1,n_member
      ie = Member(i).nm_element
```

分散並列型動的解析システム

```
      subroutine Add_damp2_ld_pa(n_member1,n_member2,nx,ld_point,Member,
     *       past_disp_point,past_vel_point,past_acc_point,
     *       am_member,Newmark_P,Element,load_mass)
       .
      do i=n_member1,n_member2
      ie = Member(i).nm_element
```

まず、簡単な変更となっているグループについて、Parallel_sf3st フォルダー内のサブルーチンをまとめると以下のようである。これらのサブルーチンは、全て部材ごとに処理を行うため、担当部材の部分を変更することになる。従って、変更は比較的容易であろう。

Add_damp2_ld_pa.for
Add_damp3_ld_pa.for
Add_earth2_ld_pa.for
Add_fdd_ld_pa.f
Add_stiff1_ld_pa.for
Add_stiff2_ld_pa.f
Add_tan_stiff_ld_pa.f
Cal_stress_pa.f
Check_Maxwell_stress_pa.f

```
Check_stress_pa.f
Get_max_stress_pa.for
Get_nonlinear_stiff_pa.f
Get_pointforce_ld_pa.for
```

上記のサブルーチンは、全て先に示した変更のみである。ただし、プログラム内で 2 箇所変更部分のサブルーチンもある。変更する場合は、注意して変更されたい。

次に、二つ目の分類は、解析制御情報を送受信するために変更を加えているサブルーチンである。マスター側では、ファイルより入力したデータをパックするためのコードを付け加えており、スレーブ側では、受信したデータを制御用パラメータに割り付けるコードを加えている。下に示すサブルーチンがこれに該当しており、プログラム名の末尾についている「_pa」はマスター側で使用し、「_set」はスレーブ側で使用する。これらのサブルーチンの説明は、既に第 4 章で行っている。変更する場合は、ここを参照されたい。

```
damctl_pa.for
damctl_set.for
doutcl_pa.for
doutcl_set.f
dyctl1_pa.for
dyctl1_set.for
dyctl2_pa.for
dyctl2_set.for
```

次に分類されるサブルーチンは、解析用データを入力し、そのデータをパックするためのサブルーチンとファイバー断面の解析データを出力するか否かをチェックするサブルーチン out_section_check_pa.f である。サブルーチン自身も比較的長くて、複雑となっている。どこが並列用に変更になっているか良く調査してから変更されたい。説明は、第 4 章で行っている。

```
Fiber_input_pa.for
Get_imperfection_pa.for
Get_structure_pa.f
out_section_check_pa.f
set_structure.for
```

以上のサブルーチンは、全て、Parallel_sf3st フォルダーに保存されている。これ以降のサブルーチンは、Parall_for フォルダーに保存されており、これらのサブルーチンも上のサブルーチン群と同様に、デ

ータの送受信で変更になっているサブルーチンである。変更箇所を良く検討した後、変更されたい。このフォルダーには、他のサブルーチンも保持されており、これらについても変更が必要かどうかチェックされたい。

```
recv_Fiber.for
recv_Fiber_P.for
recv_imperfection.for
recv_max_stress.for
recv_RO_data.for
recv_structure.for
RO_data_input_pa.for
```

最後に、フォルダーmaster と slave に、各々動的ソルバーの主サブルーチンが保存されている。主サブルーチンが変更する必要がある場合は、十分に調査してから変更されたい。

8.4 部材モデルの組み込み

本節では、変更方法を具体的に学ぶために、模擬的に新しい部材モデルを組み込むことにしよう。部材モデルは、マニュアル動的解析編で紹介している静的縮合モデルである。詳細はこの動的解析編の第 9 章を参照されたい。SPACE の単一 CPU バージョンには、既にこの部材モデルが組み込まれているものする。この部材モデルを組み込んだ SPACEVer.2.2 に対応する分散並列型動的解析システムをバージョン 1.10 と呼ぶことにする。

最新バージョンは、スレーブ制御方法を変更した Ver.1.11 である

このモデルを組み込むとき問題となるのは、モデル設定用データの送信と構造データの中の要素データで、特殊断面（ファイバー）データへのリンク情報である。問題点は、両者共にそのデータ数が不定であることにあり、新たに送受信用サブルーチンを設計しなければならない。

変更は、まず、動的ソルバーの主サブルーチンが保存されているファイル submain_pa.f について行う。ただし、このサブルーチンはマスター用とスレーブ用があるので、両者共に変更する必要がある。変更箇所は、以下のようである。

1. 新ヘッダーファイル New_submain.h を挿入する。
2. 新しく定義した動的領域を宣言・確保・解放処理のコードを挿入する。
3. 主サブルーチン内で、引数の追加があったサブルーチンは変更する。

新しく設計した以下のサブルーチンをプロジェクトに追加する。

```
Cal_check_stiff_Mx()
Cal_lin_stiff_Mxx()
Cal_nonlin_stiff_Mxx()
Cal_stress_Mx()
Fiber_checkx()
Fiber_Model_Glx()
Fiber_Model_Gx()
Get_S_comp_model()
set_modelx_dat()
Stiff_Mx()
Stiff_Mx_l()
```

次に、以下のサブルーチンの内容を変更する。このサブルーチンはマスター用であり、多くの変更点があるので注意して変更しなければならない。

```
１．Fiber_input_pa()
２．Get_structure_pa()
```

同じく、以下のサブルーチンは、スレーブ用で、ここでも多くの変更点がある。

```
１．recv_Fiber()
２．set_structure()
```

次に、前節で指摘したパラレル用のサブルーチンをチェックし、変更の有無を確かめた後、変更する。これらのサブルーチンでは、変更点はそれほど複雑ではないので、容易に変更できよう。

```
Add_damp2_ld_pa.for
Add_damp3_ld_pa.for
Add_earth2_ld_pa.for
Add_fdd_ld_pa.f
Add_stiff1_ld_pa.for
Add_stiff2_ld_pa.f
Add_tan_stiff_ld_pa.f
Cal_stress_pa.f
Check_Maxwell_stress_pa.f
Check_stress_pa.f
Get_max_stress_pa.for
Get_nonlinear_stiff_pa.f
Get_pointforce_ld_pa.for
```

```
Get_imperfection_pa.for
out_section_check_pa.f
set_structure.for
recv_Fiber_P.for
recv_imperfection.for
recv_max_stress.for
recv_RO_data.for
recv_structure.for
RO_data_input_pa.for
```

以上で、変更すべきサブルーチンが特定された。実際の変更は、単純ではあるが、注意深く行う必要がある。

8.4.1 構造体の確保とモデル設定用データの送受信

前節で示したように、新規の部材モデルを組み込むために、第 2 章から 4 章までに説明した分散並列型動的解析システムに、変更を加える必要がある。特に問題となるのは、マスター側ではモデル設定用データの送信と構造データの中の要素データから特殊データのリンク情報の送信、スレーブ側ではそれらのデータ受信とリンク情報の確立である。両者共にそのデータ数が不定であるため、新たに送受信用サブルーチンを設計しなければならない。そこで、本節では先に説明したコントロール情報の送信用仕様を変更し、それに即したサブルーチンを新たに開発する。

スレーブ側では、モデル設定用データを受信するために、動的領域を確保しなければならない。そのため、この動的領域の大きさを先にスレーブに送信しておく必要がある。ここでは、以下のようにコントロール情報の最初の部分に、この動的領域の大きさと構造体 E_Fiber_work の大きさを挿入する。

```
      id(1)=N_analysis
      id(2)=Parameter_C.n_point            !node
      id(3)=Parameter_C.n_element          !nelem
      id(4)=Parameter_C.n_member           !memb
      id(5)=Parameter_C.n_boundary_p       !nrbound
      id(6)=Parameter_C.n_local_coord      !locod
      id(7)=Parameter_C.n_rot_axis         !njiku
      id(8)=Parameter_C.n_free             !6
c                                  新規モデルのため追加
      id(9)=Parameter_C.n_S_comp_model
      id(10)=Parameter_C.nE_New_Element
```

コントロール情報のスレーブ送信用仕様を拡張する

送信用コントロール情報の 9 番目にモデル設定用データ、10 番目に構造体 E_Fiber_work の動的領域の大きさをセットしている。送信及び

受信データの大きさを管理する配列 iif_dt を、以下のように整数データの初期設定で 10 にセットする。

```
c―――― ★★解析制御情報をファイルから入力し、スレーブに転送するためバッファにセット
c――――――――――――――――――――――――――――――――★解析制御情報をファイルから入力(vpp)
      ALLOCATE (ff_data(400),if_data(400),iif_dt(2))
      iif_dt(1) = 0    ! ff_data の最終場所
      iif_dt(2) = 10   ! if_data の最終場所
```

マスター側での送信処理は、次のようである。

```
c――――――――――――――――――――――――――――――――――――――――――――――――
c
c    ★         新配列の大きさを動的確保する
c
c――――――――――――――――――――――――――――――――――――――――――――――――
      N= Parameter_C.n_S_comp_model      ! 任意静的縮合型モデル
      if(N.ne.0) then                                                  ! 1
      nx=1
      call Send_S_comp_model(nx,nx_file,S_comp_model,Model_type ,ieerx,
     *                      id_buf,jd_buf,buf,ibuf)                    ! 2
      if(ieerx.ne.0) then
      write(76,*) ' 任意静的縮合型モデル用ファイルがありません ',ieerx
      Parameter_C.n_S_comp_model=0
     return
     else
      N = Parameter_C.nE_New_Element     !  任意型静的縮合モデルに含まれる要素エレメント数
      if(N.ne.0)then                                                    ! 3
      ALLOCATE (E_Fiber_work(N))         ! 新規縮合モデルの要素エレメント数の動的確保  ①-2
      endif
      N = nx_file                        ! 任意静的縮合型モデル
      Parameter_C.n_S_comp_model=N                                      ! 4
      if(N.ne.0)then                                                    ! 5
      ALLOCATE (S_comp_model(N))         ! 新規縮合モデル設定領域の動的確保
      endif
      nx=2
      call Send_S_comp_model(nx,nx_file,S_comp_model,Model_type ,ieerx,  ! 6
     *                      id_buf,jd_buf,buf,ibuf)
      ALLOCATE (buf(id_buf+10),ibuf(jd_buf+10))                         ! 7
      nx=3
      call Send_S_comp_model(nx,nx_file,S_comp_model,Model_type ,ieerx,  ! 8
     *                      id_buf,jd_buf,buf,ibuf)
      DEALLOCATE (buf,ibuf)                                             ! 9
      endif
      endif
```

1. 新規モデルを使用しているかどうか、チェックする。使用している場合は以下の処理を行う。使用していない場合は、新規モデルがゼロであることをスレーブに送信する。
2. モデル設定用ファイルを読み込み、モデル設定用構造体の動的領域

の大きさを取得する。このファイル入力用サブルーチンについては後節で説明する。

3. 新規モデルの要素エレメントを保存する E_Fiber_work 構造体に関する動的領域を確保する。
4. 新規モデル設定用構造体の大きさを示す nx_file を構造体要素にセットする。
5. 新規モデルを設定する構造体 S_comp_model の動的領域を確保する。
6. 2 回目のモデル設定用ファイルを読み込み、マスター用の構造体にセットする。さらに、送信用のバッファ領域である動敵的域の大きさを数える。これは、このモデル設定用ファイル内のデータ数が不定であるため、その数を数える必要がある。
7. バッファ領域である ibuf と buf の動的領域を確保する。
8. 3 回目のモデル設定用ファイルを読み込み、送信用のバッファ領域にデータをセットする。最後に、そのバッファ内のデータを全てのスレーブに送信する。
9. 送信用バッファ領域を解放する。

スレーブ側では、次のサブルーチンコールによって、コントロール情報を取得する。

```
c―――――― ★★解析制御情報をマスターから取得
      write(*, *) "enter recv_ctlset  xxxxxxxxx"
c―――――― ★★解析制御情報をマスターから取得(vpp)
      call recv_ctlset(Parameter_C,ff_data,if_data,iif_dt,N_analysis,
     *                 MPP_Ana_Group,ierr_dat)
      write(*, *) "out recv_ctlset : Parameter_C.n_element =",
     *                              Parameter_C.n_element
```

ここでは、先に示した Parameter_C の要素である n_S_comp_model と nE_New_Element に 2 つの情報をセットしている。この 2 つの情報を用いて、スレーブ側では以下の処理を行い、新規モデル設定用データを取得する。

```
c―――――――――――――――――――――――――――――――――――――――――――――――――――――
c
c     ★          新配列の大きさを動的確保する
c
c―――――――――――――――――――――――――――――――――――――――――――――――――――――
      N= Parameter_C.n_S_comp_model       ! 任意静的縮合型モデル
      if(N.ne.0) then                                                  ! 1
      N = Parameter_C.nE_New_Element      !  任意型静的縮合モデルに含まれる要素エレメント数
      if(N.ne.0)then                                                   ! 2
      ALLOCATE (E_Fiber_work(N))          ! 新規縮合モデルの要素エレメント数の動的確保
```

```
      endif
      nx=1
      call recv_S_comp_model(nx,nx_file,S_comp_model,Model_type ,ieerx,
     *                       id_buf,jd_buf,buf,ibuf)                          ! 3
      Parameter_C.n_S_comp_model=nx_file
      ALLOCATE (S_comp_model(nx_file))          ! 新規縮合モデル設定領域の動的確保    4
      ALLOCATE (buf(id_buf+10),ibuf(jd_buf+10))                              ! 5
      nx=2
      call recv_S_comp_model(nx,nx_file,S_comp_model,Model_type ,ieerx,
     *                       id_buf,jd_buf,buf,ibuf)                          ! 6
      DEALLOCATE (buf,ibuf)                                                  ! 7
```

1. 新規モデルを使用しているかどうか、チェックする。使用している場合は以下の処理を行う。
2. 新規モデルの要素エレメントを保存する E_Fiber_work 構造体に関する動的領域を確保する。
3. サブルーチン recv_S_comp_model()を用いて、新規モデル設定用の制御データを取得し、その中の新規モデル設定用構造体の大きさを示す nx_file を構造体要素にセットする。
4. 新規モデルを設定する構造体 S_comp_model の動的領域を確保する。
5. バッファ領域である ibuf と buf の動的領域を確保する。
6. モデル設定用データを受信し、スレーブ用の構造体にセットする。
7. 送信用バッファ領域を解放する。

8.4.2 新規モデルの特殊データの送受信

新規モデルの組み込みによって、もうひとつの情報を特別に送信する必要がある。これは構造データファイル内の要素データで、新規モデルの第 2 レコードはモデルの設定にしたがうために、その数は不定となる。そのため、一般の構造データ送信用ルーチンではデータ送信を実行することができない。そこで、その部分のみ、特別に send_E_fiber()サブルーチンを用いて送信することになる。送信の有無は、新規モデル設定用ファイルの動的領域の大きさを表す Parameter_C.n_S_comp_model の値によって決める。無論、並列処理を行うときのみ、この処理は行われる。上記処理が終了した後、一般の構造用データを全スレーブに送信する。

```
c------------------------------------------------------------------
c
c     ★          E_Fiber_work の内容を別途送信
c
c------------------------------------------------------------------
```

```
      N= Parameter_C.n_S_comp_model       ! 任意静的縮合型モデル
c     write(76,*) ' Parameter_C.n_S_comp_model ',n
      if(n_proc.ge.2) then
      if(N.ne.0) then
      call send_E_fiber(E_Fiber_work,S_comp_model,
     *                                Element,Parameter_C)
      endif
c―――――― ★★　スレーブ、モニターに構造データを送信
      call Send_structure(buf,ibuf,id_buf,jd_buf)
      endif
      DEALLOCATE (buf,ibuf)
```

スレーブ側の受信コードは、以下のようである。

```
c―――――――――――――――――――――――――――――――――――――――――――――――――――――――
c
c     ★         E_Fiber_work の内容を別途送信
c
c―――――――――――――――――――――――――――――――――――――――――――――――――――――――
      call recv_E_fiber(E_Fiber_work )
      endif
c―――――――――――――――――――――――――――――――――――――――――――――――――――――――
c
c     ★         配列の大きさを動的確保する
c
c―――――――――――――――――――――――――――――――――――――――――――――――――――――――
      N=max_member              ! 担当部材数
      ALLOCATE (
     *   am_member(12,12,N),
     *   rot_memb_t(3,3,N),rot_memb(3,3,2,N),
     *   ak_linear(12,12,N),ak_nonlinear(12,12,N)
     *         )
      N=Parameter_C.n_local_coord       ! 局所座標系を使用する場合
      if(N.ne.0)ALLOCATE (
     *   rot_local(3,3,N)
     *)
      M_alloc(1)=1
c―――――――――――――――――――――――――――――――――――――――――――――――――――――――
c
c     ★         構造・荷重データをマスターから取得する
c
c―――――――――――――――――――――――――――――――――――――――――――――――――――――――
c――――――――――――――――――――――――――――――――――――――――――――――――――――――★基本構造データをマスターから取得(vpp)
c―――――― ★★マスターから構造データを取得
c     バッファデータ仕様
c     buf()     １：座標 3
c               ２：局所座標 3
c               ３：要素 17
c               ４：部材 4
c     ibuf()    １：境界拘束条件 7
c               ２：要素 6
c               ３：部材  14
c
```

```
       id_buf=Parameter_C.n_point*6+Parameter_C.n_element*17+
      *       Parameter_C.n_member*4
       jd_buf=Parameter_C.n_point*7++Parameter_C.n_element*6+
      *       Parameter_C.n_member*14                    ! 全部材数用意する
       ALLOCATE (buf(id_buf+10),ibuf(jd_buf+10))
c      write(*, *) ' allocate ',id_buf, jd_buf
c―――――― ★★構造データ受信
       call recv_structure(buf,ibuf,id_buf,jd_buf)
c      write(*, *) ' recev_structure ok'
       call set_structure(Point,Member,Element,Parameter_C,
      *       Model_type,ierr,buf,ibuf,n_member1,n_member2,
      *       S_comp_model,E_Fiber_work)
c      write(*, *) ' set structure ok',max_member
       DEALLOCATE (buf,ibuf)
      Parameter_C.n_member = max_member     !スレーブ側で計算する担当部材数に変換
```

ここでは、新規モデルを使用している場合はデータを受信し、構造体にセットする。次は、バッファ領域を動的に確保し、サブルーチン recv_structure()で一般の構造データを受信する。このバッファを用いて、サブルーチン set_structure()で構造体にセットする。

この中で、サブルーチン send_E_fiber()と recv_E_fiber()について説明しよう。これらのサブルーチンの内容を以下に示す。

```
C     ―――――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /send_E_fiber
C       ―――――――――――――――――――――――――――――――――――――――
C     ●   静的縮合部材モデルのリンク情報を送信する
C     ―――――――――――――――――――――――――――――――――――――――
       subroutine send_E_fiber(E_Fiber_work,S_comp_model,
      *                                Element,Parameter_C)
       use MPI_DEFINE
       implicit real*8(A-H,O-Z)
       include "..¥..¥sf3st¥submain.h"
       include "..¥..¥sf3st¥New_submain.h"
       record / E_Fiber_work_s / E_Fiber_work
       record / S_comp_model_s / S_comp_model
       record / parameter_s    / Parameter_C
       record / element_s      / Element
       dimension S_comp_model(*),E_Fiber_work(*)
       dimension Element(*)
       integer, save, ALLOCATABLE  ::ibuf(:)

       jd_buf=0
       do i=1,Parameter_C.n_element                                    ! 1
       m_type = Element(i).element_type
       mmx=(m_type-1)/10
       if(mmx .eq. 5 .or. mmx .eq. 6) then                             ! 2
       nxx=S_comp_model(m_type-50).n_div_element                       ! 3
       jd_buf=jd_buf+nxx
       endif
       enddo
```

```
       ALLOCATE (ibuf(jd_buf+10))                                    ! 4
      do j=1,jd_buf
      ibuf(j) = E_Fiber_work(j).n_Fiber_section                      ! 5
      enddo
c      write(76,'(10i6)') (ibuf(j),j=1,jd_buf)
c──────── ★★  制御情報と構造用制御情報を受信        ★★────────
      call MPI_Bcast(jd_buf,1, MPI_INTEGER,
     *               ID_MASTER, MPI_COMM_WORLD, ierr)               ! 6
      call MPI_Bcast(ibuf,jd_buf, MPI_INTEGER,
     +               ID_MASTER, MPI_COMM_WORLD, ierr)               ! 7
c──────── ★★  ──────────────────────────── ★★────────
      DEALLOCATE (ibuf)                                             ! 8
      return
      end
```

1. 全要素について、以下の処理を行う。
2. 新規モデルであるかチェックする。
3. 新規モデルの場合、そのモデルのエレメント数を取り出し、その値を全要素、ただし新規モデルについて和を取る。
4. 新規モデルの全エレメント数 jd_buf を用いて、バッファ用動的領域を確保する。
5. 動的確保したバッファ領域にエレメントの特殊断面（ファイバー断面番号）へのリンク情報をセットする。
6. 最初に送信個数を全スレーブに送信する。
7. 次に、バッファ領域のデータを全スレーブに送信する。
8. 送信用バッファ領域を解放する。

次に、スレーブ側の受信用サブルーチンを以下に示す。

```
C     ──────────────────────────────────────────────
C     ●   SUBROUTINE /recv_E_fiber
C       ──────────────────────────────────────────────
C     ●   静的縮合部材モデルのリンク情報を受信する
C     ──────────────────────────────────────────────
      subroutine recv_E_fiber(E_Fiber_work)
      use MPI_DEFINE
      implicit real*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
      include "..¥..¥sf3st¥New_submain.h"
      record / E_Fiber_work_s / E_Fiber_work
      dimension E_Fiber_work(*)
      integer, save, ALLOCATABLE  ::ibuf(:)
      dimension iibuf(5)
c──────── ★★  リンク情報を受信する              ★★────────
      write(76,'(a)') '  recv e_fiber'
      call MPI_Bcast(jd_buf,1, MPI_INTEGER,                         ! 1
```

```
      *                ID_MASTER, MPI_COMM_WORLD, ierr)
       ALLOCATE (ibuf(jd_buf+10))                                        ! 2
       call MPI_Bcast(ibuf,jd_buf, MPI_INTEGER,
      +                ID_MASTER, MPI_COMM_WORLD, ierr)                 ! 3
c―――― ★★ ―――――――――――――――――――――――――― ★★――――
       do j=1,jd_buf                                                     ! 4
       E_Fiber_work(j).n_Fiber_section =ibuf(j)                          ! 5
       enddo
       DEALLOCATE (ibuf)                                                 ! 6
       return
       end
```

1. 最初に、受信する新規モデル設定用データの受信個数 jd_buf を受信する。
2. 受信個数 jd_buf を用いて、受信バッファの動的領域を確保する。
3. 新規モデル設定用データを受信する。
4. 受信個数 jd_buf、つまり新規モデルの総エレメント数分、以下の処理を行う。
5. 受信データを構造体にセットする。これで、マスターと同一の状態を作り出すことができる。
6. 受信用バッファ領域を解放する。

8.4.3 コントロールデータの送受信

本節では、コントロールデータの送受信を行うサブルーチンで、変更する部分について説明する。最初に、コントロールデータを全スレーブに送信するサブルーチン send_ctlset()を以下に示す。

```
C     ―――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /send_ctlset
C       ―――――――――――――――――――――――――――――――――――
C     ●   制御情報をスレーブに転送する(ok)
C     ―――――――――――――――――――――――――――――――――――
      subroutine send_ctlset(Parameter_C,fd,id,iif_dt, N_analysis)
c     外部宣言
      use MPI_DEFINE
       IMPLICIT REAL*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
c     引数
       record /parameter_s     / Parameter_C! 構造基本データ
      real*4 fd(100)
      integer :: id(100), iif_dt(2)
c     実装
       id(1)=N_analysis
       id(2)=Parameter_C.n_point            !node                        ! 1
```

```
      id(3)=Parameter_C.n_element          !nelem
      id(4)=Parameter_C.n_member           !memb
      id(5)=Parameter_C.n_boundary_p       !nrbound
      id(6)=Parameter_C.n_local_coord      !locod
      id(7)=Parameter_C.n_rot_axis         !njiku
      id(8)=Parameter_C.n_free             !6
c                              新規モデルのため追加
      id(9)=Parameter_C.n_S_comp_model                                    ! 2
      id(10)=Parameter_C.nE_New_Element                                   ! 3
c      write(76,'(a,2i5)') ' 制御データ',iif_dt(2),iif_dt(1)
c     write(76,'(10i8)') (id(i),i=1,iif_dt(2))
c     write(76,'(10f10.2)') (fd(i),i=1,iif_dt(1))
c------- ★★  制御情報と構造用制御情報を送信        ★★-------
      call mpi_bcast(iif_dt, 2, MPI_INTEGER,ID_MASTER,                    ! 4
     +                                  MPI_COMM_WORLD,ierr)
      call mpi_bcast(id,iif_dt(2),MPI_INTEGER,ID_MASTER,                  ! 5
     +                                  MPI_COMM_WORLD,ierr)
      call mpi_bcast(fd,iif_dt(1),MPI_REAL,ID_MASTER,                     ! 6
     +                                  MPI_COMM_WORLD,ierr)
c------- ★★  ---------------------------------- ★★-------
      RETURN
      END
```

1. コントロールデータの整数動的領域 id()の先頭部分の８つに、解析用のデータをセットする。
2. 同じく整数動的領域 id()の 9 番目に、新規モデルの動的領域個数をセットする。新規モデルを使用していない場合は 0 がセットされる。
3. 新規モデルのエレメント数を構造体から取り出し、その値を整数動的領域 id()の 10 番目にセットする。
4. 次に送信する実数データ数と整数データ数を、まず、送信する。
5. 整数データを送信する。
6. 実数データを送信する。

次は、受信用サブルーチン recv_ctlset()の内容を以下に示す。

```
C     ------------------------------------------------------------
C     ●   SUBROUTINE /recv_ctlset
C     ------------------------------------------------------------
C     ●   制御ファイルをマスターより受信(ok)（スレーブ用）
C     ------------------------------------------------------------
      subroutine recv_ctlset(Parameter_C,ff_data,if_data,iif_dt,
     *                       N_analysis,MPP_Ana_Group,ierr_dat)
c 外部宣言
      use MPI_DEFINE
      IMPLICIT REAL*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
c  引数宣言
```

```
      record /parameter_s    / Parameter_C
     integer:: if_data(*), iif_dt(2,4)
      real*4:: ff_data(*)
c  内部変数
      integer:: ii_dt(2)
c  本体
c――――― ★★  制御情報と構造用制御情報を受信       ★★―――――
      call MPI_Bcast(ii_dt, 2, MPI_INTEGER,
     +               ID_MASTER, MPI_COMM_WORLD, ierr)                 ! 1
      call MPI_Bcast(if_data, ii_dt(2), MPI_INTEGER,
     +               ID_MASTER, MPI_COMM_WORLD, ierr)                 ! 2
      call MPI_Bcast(ff_data, ii_dt(1), MPI_REAL,
     +               ID_MASTER, MPI_COMM_WORLD, ierr)                 ! 3
c――――― ★★ ――――――――――――――――――――――― ★★―――――
      N_analysis          = if_data(1)                                ! 4
      Parameter_C.n_point = if_data(2)             !node
      Parameter_C.n_element = if_data(3)           !nelem
      Parameter_C.n_member  = if_data(4)           !memb
      Parameter_C.n_boundary_p  = if_data(5)       !nrbound
      Parameter_C.n_local_coord  = if_data(6)      !locod
      Parameter_C.n_rot_axis  = if_data(7)         !njiku
      Parameter_C.n_free  = if_data(8)             !6
c                     新規モデルのため追加
      Parameter_C.n_S_comp_model  = if_data(9)                        ! 5
      Parameter_C.nE_New_Element  = if_data(10)                       ! 6
      write(*,'(10i5)') (if_data(i),i=1,10)
      iif_dt(1,1)=1                                                   ! 7
      iif_dt(2,1)=11                                                  ! 8
c―――――――――――――――――――――――――――――――――――★制御データの先頭番地をセット
      do i=1,3
      i1=iif_dt(1,i)
      i2=iif_dt(2,i)
      iif_dt(2,i+1)=iif_dt(2,i)+if_data(i2)+1                         ! 9
      iif_dt(1,i+1)=iif_dt(1,i)+ff_data(i1)+1                         ! 10
      enddo
      RETURN
      END
```

1. 最初に、実際に受信する実数データ数と整数データ数を受信する。
2. 上記の値を用いて、整数データを受信する。
3. 上記の値を用いて、実数データを受信する。
4. コントロールデータの整数動的領域 id()の頭の部分から、解析用データとして該当する構造体にセットする。
5. 同じく整数動的領域 id()の 9 番目の値を、新規モデルの動的領域個数を表す構造体 Parameter_C.n_S_comp_model にセットする。
6. 整数動的領域 id()の 10 番目の値を、新規モデルのエレメント数を表す構造体 Parameter_C.nE_New_Element にセットする。
7. 最初のダイアログに関する実数データの初期値をセットする。

8. 最初のダイアログに関する整数データの初期値をセットする。
9. 第 i 番目のダイアログの整数データ個数を加える。
10. 第 i 番目のダイアログの次数データ個数を加える。

ふたつのバッファ領域には、各解析をコントロールするダイアログで定義したデータと共に、その最初の部分に当該のデータ個数がセットされている。その値を用いて上記の処理によって、制御データの先頭番地を配列 iif_dt()にセットする。

8.4.4 モデル設定用データの送受信

本節では、モデル設定用データを送受信するサブルーチンを説明する。このサブルーチン Send_S_comp_model()は 3 回コールされ、モデル設定用ファイルが 3 度読み込まれる。流れの制御は引数 nx で行われる。1 回目は、モデル設定用ファイルの制御データを読み込み、構造体の大きさを決定するためのデータを取得する。2 回目も、同じくモデル設定用ファイルを読み込み、マスター用の構造体にデータをセットする。その際、データ送信のためのバッファ領域の大きさを数える。3 回目は、再度、このモデル設定用ファイルを読み込み、バッファ領域にデータをセットする。全てデータをセットした後、スレーブにデータを送信する。

次に、送信用のサブルーチン Send_S_comp_model()の内容を以下に示す。なお、モデル設定用ファイルの仕様は、コメント行に書き込まれているので参照されたい。

```
C     ―――――――――――――――――――――――――――――――――――――――
C     ●    SUBROUTINE /Send_S_comp_model
C     ―――――――――――――――――――――――――――――――――――――――
C     ●    静的縮合部材モデルの定義ファイルを読み、送信する
C     ―――――――――――――――――――――――――――――――――――――――
      subroutine Send_S_comp_model(nx, nx_file,S_comp_model,
     *                             Model_type,ieerx, id_buf,jd_buf,buf,ibuf)
      use MPI_DEFINE
      implicit real*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
      include "..¥..¥sf3st¥New_submain.h"
      record / n_model_s          / Model_type
      record / S_comp_model_s     / S_comp_model
      dimension S_comp_model(*)
      character aa*80
      real*8:: buf(*)
      integer :: ibuf(*)
      integer :: id_buf, jd_buf
      integer:: iibuf(5)
C     ―――――――――――――――――――――――――――――――――――――――
```

```
c     nx              :1: パラメータセット 2:データ入力
c     nx_file     設定モデルの最大モデル番号 - 50 ：構造体 S_comp_model の確保用
c     1. コメント行数
c     2. 上記行数分、全体コメント
c     3. モデル個数、最大モデル番号
c        以下のデータをモデル個数分繰り返す
c     4. 1 行のモデル用コメント
c     5. モデル番号、部材モデル中の要素数(n_div)
c     6. 要素モデル番号(1 - n_div) x 軸原点より順に設定する。
C     7. 要素の長さ (1 - n_div) (部材長さを 1. として、比率で設定する)
C     8. 次のデータを (1 - n_div+1) 繰り返す
C     9. 節点自由度(1 - 6 )  x 軸原点より順に設定する。
C     10.弾塑性応力出力・表示番号 (1 -  n_out_stress)
C     11.応力出力数、応力出力番号(1-5)
C     ------------------------------------------------------------------
      ierrx=0
      if(nx.eq.1) then                                                ! 1
C     ------------------------------------------------------------------
c     ★          モデルの定義ファイルを予備入力し、構造体の値を設定する
C     ------------------------------------------------------------------
      nfix=5
      nfi=65
      call infile(nfi,nfix,ierr)                                      ! 2
      if(ierr.eq.0) then
      read(nfix,*) ii                                                 ! 3
      do i=1,ii
      read(nfix,*)  aa
      enddo
      read(nfix,*) nn,nx_file                                         ! 4
      nx_file=nx_file-50                                              ! 5
      close(nfix)
      else
      ierr=1
      endif
      return                                                          ! 6
C     ------------------------------------------------------------------
c     ★          構造体にデータをセットし、バッファ領域の大きさを数える
C     ------------------------------------------------------------------
      elseif(nx.eq.2) then                                            ! 7
      nfix=5
      nfi=65
      i_int=0
      i_real=0
      call infile(nfi,nfix,ierr)                                      ! 8
      read(nfix,*) ii
      write(76,'(//a,i4)') ' 静的縮合モデルの定義ファイル '
      do i=1,ii
      read(nfix,*)  aa
      write(76,'(10x,a80)')  aa
      enddo
      i_int=i_int+1
      read(nfix,*) nn, nx_file
      write(76,'(/2i4)') nn, nx_file
      nx_file=nx_file-50
```

```
      do ij=1, nn                                                    ! 9
      read(nfix,*)  aa                                               ! 10
      write(76,'(//10x,a80)') aa
      read(nfix,*)  number, n_div                                    ! 11
      i_int=i_int+2                                                  ! 12
      i= number -50                              ! モデルより 50 を減ずる
      write(76,'(3i4)') i,number, n_div
      S_comp_model(i).n_div_element = n_div                          ! 13
      i_int=i_int+n_div                                              ! 14
      read(nfix,*) (S_comp_model(i).nm_type_element(j),j=1,n_div)    ! 15
      write(76,'(10i4)') (S_comp_model(i).nm_type_element(j),j=1,n_div)
      i_real=i_real+n_div                                            ! 16
      read(nfix,*) (S_comp_model(i).alength(j),j=1,n_div)            ! 17
      write(76,'(10f12.4)') (S_comp_model(i).alength(j),j=1,n_div)
      i_int=i_int+(n_div+1)*6                                        ! 18
      do j=1,n_div+1                                                 ! 19
      read(nfix,*) (S_comp_model(i).irest_Point(k,j),k=1,6)          ! 20
      write(76,'(10i4)')j, (S_comp_model(i).irest_Point(k,j),k=1,6)
      enddo
      i_int=i_int+n_div                                              ! 21
      read(nfix,*) (S_comp_model(i).nm_out_stress(j),j=1, n_div)     ! 22
      write(76,'(10i4)') (S_comp_model(i).nm_out_stress(j),
     *       j=1, n_div)
      n_out_stress=0
      do j=1,n_div                                                   ! 23
      if(n_out_stress.lt.S_comp_model(i).nm_out_stress(j))
     +      n_out_stress=S_comp_model(i).nm_out_stress(j)
      enddo
      S_comp_model(i).nm_out_stress_x=n_out_stress                   ! 24
      i_int=i_int+6
      read(nfix,*) n_out_stress,                                     ! 25
     *     (S_comp_model(i).n_out_stress_x(j),j=1, 5)
      write(76,'(10i4)') n_out_stress,
     *     (S_comp_model(i).n_out_stress_x(j),j=1, 5)
      S_comp_model(i).n_out_stress=n_out_stress
C     ------------------------------------------------------------------
      Model_type.no_e_model(number)   = number  !要素モデルの番号      ! 26
      Model_type.n_div_model(number)  = n_div   !要素モデルの分割数
      Model_type.n_e_model(number)    = 0       !要素モデルの数
      Model_type.n_m_model(number)    = 0       !部材モデルの数
      Model_type.n_damp(number)       = 0       !部材減衰ありか
C     ------------------------------------------------------------------
c     ★          各要素の個数を数える
C     ------------------------------------------------------------------
      S_comp_model(i).i_set_ok=0                                     ! 27
      do k=1,50
      S_comp_model(i).n_type_element(k)=0
      enddo
      do k=1,n_div                                                   ! 28
      j= S_comp_model(i).nm_type_element(k)
      S_comp_model(i).n_type_element(j)=                             ! 29
     *         S_comp_model(i).n_type_element(j)+1
      enddo
c      write(76,*)(S_comp_model(i).n_type_element(j),j=1,50)
```

```
      enddo
      close(nfix)                                                    ! 30
      id_buf=i_real                                                  ! 31
      jd_buf=i_int
C     ------------------------------------------------------------------
c     ★          バッファ領域にデータをセットする
C     ------------------------------------------------------------------
      Else                                                           ! 32
      i_real=0
      i_int=0
      nfix=5
      nfi=65
      call infile(nfi,nfix,ierr)
      read(nfix,*) ii
      do i=1,ii
      read(nfix,*)  aa
      enddo
      i_int=i_int+1
      read(nfix,*) nn, nx_file
      ibuf(i_int)=nn
      nx_file=nx_file-50
      do ij=1, nn
      read(nfix,*)  aa
      read(nfix,*)  number, n_div
      i_int=i_int+1
      ibuf(i_int)=number
      i_int=i_int+1
      ibuf(i_int)=n_div
      i= number -50                                  ! モデルより 50 を減ずる
c     write(76,'(3i4)') i,number, n_div
c      S_comp_model(i).n_div_element = n_div
      read(nfix,*) (ibuf(i_int+j),j=1,n_div)
      i_int=i_int+n_div
      read(nfix,*) (buf(i_real+j),j=1,n_div)
      i_real=i_real+n_div
      do j=1,n_div+1
      read(nfix,*) (ibuf(i_int+k),k=1,6)
      i_int=i_int+6
      enddo
      read(nfix,*) (ibuf(i_int+j),j=1,n_div)
      i_int=i_int+n_div
      read(nfix,*) (ibuf(i_int+j),j=1,6)
      i_int=i_int+6
      enddo
      close(nfix)

c------- ★★  制御情報と構造用制御情報を受信        ★★-------
      iibuf(1)=id_buf                                                ! 33
      iibuf(2)=jd_buf
      iibuf(3)=nx_file
      call mpi_bcast(iibuf,3, MPI_INTEGER,                          ! 34
     +               ID_MASTER,MPI_COMM_WORLD,ierr)
c------- ★★  制御情報と構造用制御情報を受信        ★★-------
      call mpi_bcast(ibuf,iibuf(2), MPI_INTEGER,
```

```
     +                   ID_MASTER, MPI_COMM_WORLD, ierr)                    ! 35
      call mpi_bcast(buf,iibuf(1), MPI_DOUBLE,
     +                   ID_MASTER, MPI_COMM_WORLD, ierr)                    ! 36
c------- ★★ ---------------------------------- ★★-------
      endif
      return
      end
```

1. 1 回目のサブルーチンコールによる処理が、以降で行われる。
2. モデル設定用ファイルをオープンする。
3. 新規モデルに対する設定ファイルのタイトル行数を入力する。この行数分、タイトルを読み込む。
4. モデルの個数 nn と、ここで使用する最大モデル番号 nx_file を入力する。
5. 入力した最大モデル番号から 50 を引いて、新規モデルのデータを保存する構造体の個数を求める。その後、ファイルをクローズする。
6. 1 回目のサブルーチンコールによる処理が終了し、戻る。
7. 2 回目のサブルーチンコールによって、以降の処理を行う。
8. 同じモデル設定ファイルをオープンする。
9. 1 回目と同様にデータ入力し、変数に値をセットする。さらに、これ以降で、新規モデル数分、データ入力を繰り返す。
10. このモデルのタイトルを 1 行で入力する。
11. ここでは、2 つのデータを読み込む。最初は部材モデル番号であり、次はこのモデルのエレメント数である。
12. 送信用バッファ領域を数えるために、整数と実数の個数を数える。まず、整数個数 i_int に 2 を加える。また、モデル番号から 50 を引き、構造体 S_comp_model の引数 i をセットする。
13. 構造体 S_comp_model の要素 n_div_element にエレメント数をセットする。
14. 次のデータに関して、整数個数 i_int にエレメント数を加える。
15. 部材モデルの各エレメントの特殊断面番号、弾性部材番号等を入力し、構造体にセットする。
16. 次のデータに関して、実数個数 i_real にエレメント数を加える。
17. 部材モデルの各エレメントの長さ比率を入力し、構造体にセットする。
18. 次のデータに関して、整数個数 i_int に、6 自由度に節点数を掛けた数を加える。
19. 節点数分、以下の処理を行う。

20. その節点の自由度を入力し、構造体にセットする。
21. 次のデータに関して、次数個数 i_real にエレメント数を加える。
22. ファイバー用弾塑性応力出力用パラメータを入力し、構造体にセットする。
23. ファイバー用弾塑性応力の最大出力個数を求める。
24. 上記の値を構造体にセットする。
25. 次のデータのために、整数個数 i_int に、6 を加える。部材応力のファイル出力パラメータを入力し、構造体にセットする。
26. 部材モデル管理のために、構造体 Model_type にデータをセットする。
27. この部材モデルで、各エレメントがどのような特殊断面番号、弾性部材番号等をいくつ使用するかをチェックする。まず、ゼロセットする。
28. この部材モデルのエレメント数分、以下の処理を行う。
29. 特殊断面番号、弾性部材番号等の断面番号ごとに 1 を加える。
30. ファイルをクローズする。
31. 実数データ個数と整数データ個数をバッファ管理データにコピーする。これで、2 回目のサブルーチンコールによる処理を終了する。
32. 3 回目のサブルーチンコールの処理が以降で行われる。2 回目と同様に、新規モデル設定ファイルが読み込まれ、バッファ領域に順次セットされる。
33. 送信用実数データ個数、整数データ個数、最大構造体個数をセットする。
34. 上記で、セットした制御情報を全スレーブに送信する。
35. 整数データを全スレーブに送信する。
36. 実数データを全スレーブに送信する。以上で、新規モデルの設定に関する処理が全て終了する。

次に、このモデル設定用データを受信するためのサブルーチンを示す。

```
C     ------------------------------------------------------------
C     ●   SUBROUTINE /recv_S_comp_model
C       ----------------------------------------------------------
C     ●   静的縮合部材モデルの定義ファイルを受信する
C     ------------------------------------------------------------
      subroutine recv_S_comp_model(nx, nx_file,S_comp_model,
     *                             Model_type,ieerx,
     *                       id_buf,jd_buf,buf,ibuf)
      use MPI_DEFINE
      implicit real*8(A-H,O-Z)
```

```
      include "..¥..¥sf3st¥submain.h"
      include "..¥..¥sf3st¥New_submain.h"
      record / n_model_s       / Model_type
      record / S_comp_model_s  / S_comp_model
      dimension S_comp_model(*)
      real*8:: buf(*)
      integer:: ibuf(*)
      integer:: id_buf, jd_buf
      integer:: iibuf(5)
      character aa*80
C     ------------------------------------------------------------------
c     nx :1: パラメータセット 2:データ入力
c     nx_file     設定モデルの最大モデル番号 - 50  : 構造体 S_comp_model の確保用
c     1. コメント行数
c     2. 上記行数分、全体コメント
c     3. モデル個数、最大モデル番号
c        以下のデータをモデル個数分繰り返す
c     4. 1 行のモデル用コメント
c     5. モデル番号、部材モデル中の要素数(n_div)
c     6. 要素モデル番号(1 - n_div) x 軸原点より順に設定する。
C     7. 要素の長さ (1 - n_div) (部材長さを 1. として、比率で設定する)
C     8. 次のデータを (1 - n_div+1) 繰り返す
C     9. 節点自由度(1 - 6 )  x 軸原点より順に設定する。
C     10.弾塑性応力出力・表示番号 (1 -  n_out_stress)
C     11.応力出力数、応力出力番号(1-5)
C     ------------------------------------------------------------------
      ierrx=0
      if(nx.eq.1) then                                                  ! 1
C     ------------------------------------------------------------------
c     ★         モデルの定義ファイルを予備入力し、構造体の値を設定する
C     ------------------------------------------------------------------
c------- ★★  制御情報と構造用制御情報を受信        ★★-------
      call MPI_Bcast(iibuf,3, MPI_INTEGER,
     *               ID_MASTER, MPI_COMM_WORLD, ierr)                   ! 2
      id_buf=iibuf(1)
      jd_buf=iibuf(2)
      nx_file=iibuf(3)
      return
c------- ★★  ------------------------------------ ★★-------
C     ------------------------------------------------------------------
c     ★         静的縮合モデルの定義ファイルを入力する
C     ------------------------------------------------------------------
      else
c------- ★★  制御情報と構造用制御情報を受信        ★★-------
      call MPI_Bcast(ibuf,jd_buf, MPI_INTEGER,
     +               ID_MASTER, MPI_COMM_WORLD, ierr)                   ! 3
      call MPI_Bcast(buf,id_buf, MPI_DOUBLE,
     +               ID_MASTER, MPI_COMM_WORLD, ierr)                   ! 4
c      write(76,'(10f12.2)') (buf(j),j=1,id_buf)
c------- ★★  ------------------------------------ ★★-------
      i_int=1                                                           ! 5
      i_real=0
      nn=ibuf(i_int)                                                    ! 6
c      read(nfix,*) nn, nx_file
```

```
      do ij=1, nn
      i_int=i_int+1
      number= ibuf(i_int)
      i_int=i_int+1
      n_div= ibuf(i_int)
c      read(nfix,*)  number, n_div
      i= number -50                              ! モデルより 50 を減ずる
      S_comp_model(i).n_div_element = n_div
c      read(nfix,*) (S_comp_model(i).nm_type_element(j),j=1,n_div)
      do j=1,n_div
      i_int=i_int+1
      S_comp_model(i).nm_type_element(j)=ibuf(i_int)
      enddo
      do j=1,n_div
      i_real=i_real+1
      S_comp_model(i).alength(j)=buf(i_real)
      enddo
      do j=1,n_div+1
c      read(nfix,*) (S_comp_model(i).irest_Point(k,j),k=1,6)
      do k=1,6
      i_int=i_int+1
      S_comp_model(i).irest_Point(k,j)=ibuf(i_int)
      enddo
      enddo
c      read(nfix,*) (S_comp_model(i).nm_out_stress(j),j=1, n_div)
      do j=1,n_div
      i_int=i_int+1
      S_comp_model(i).nm_out_stress(j)=ibuf(i_int)
      enddo
      n_out_stress=0
      do j=1,n_div
      if(n_out_stress.lt.S_comp_model(i).nm_out_stress(j))
     +      n_out_stress=S_comp_model(i).nm_out_stress(j)
      enddo
      S_comp_model(i).nm_out_stress_x=n_out_stress
c      read(nfix,*) n_out_stress,
c     *     (S_comp_model(i).n_out_stress_x(j),j=1, 5)
      i_int=i_int+1
      n_out_stress=ibuf(i_int)
      do j=1,5
      i_int=i_int+1
      S_comp_model(i).n_out_stress_x(j)=ibuf(i_int)
      enddo
      S_comp_model(i).n_out_stress=n_out_stress
C     ------------------------------------------------------------------
      Model_type.no_e_model(number)   = number  !要素モデルの番号
      Model_type.n_div_model(number)  = n_div   !要素モデルの分割数
      Model_type.n_e_model(number)    = 0       !要素モデルの数
      Model_type.n_m_model(number)    = 0       !部材モデルの数
      Model_type.n_damp(number)       = 0       !部材減衰ありか
C     ------------------------------------------------------------------
c     ★          各要素の個数を数える
C     ------------------------------------------------------------------
      S_comp_model(i).i_set_ok=0
```

```
      do k=1,50
      S_comp_model(i).n_type_element(k)=0
      enddo
      do k=1,n_div
      j= S_comp_model(i).nm_type_element(k)
      S_comp_model(i).n_type_element(j)=
     *         S_comp_model(i).n_type_element(j)+1
      enddo
      enddo

      endif
      return
      end
```

1. このサブルーチンが 2 回コールされる。1 回目は以降の処理を行う。
2. 実際のデータを受信するための制御データを受信する。このデータを、バッファ領域を設定するために用いる変数にコピーする。1 回目のサブルーチンコールはこれで終了する。
3. 整数データを受信する。
4. 実数データを受信する。
5. 整数、実数共にバッファ領域の位置を示す変数を初期設定する。
6. 整数バッファ領域から全モデル数を取得する。
7. 以降の処理は、送信処理と同様であり、ファイルからデータを入力する替わりにバッファ領域からデータを取り出す。コメント行でデータ入力コードを示しており、それらを参考にするば、コードを理解することは容易である。

8.5 剛床モデル

8.5.1 はじめに

SPACE（Ver.2.20）では、立体骨組の静的解析や動的解析を行っている。その際、通常の立体骨組構造では、床の扱いをどのようにするかが問題となる。以前のバージョンでは、床をブレースに置換して解析を行っていた。ここでは、立体骨組構造で良く用いられる剛床仮定を SPACE 分散並列型動的解析システムに組み込む手法について検討する。さらに、この節では、実際のプログラムを用いて剛床仮定を SPACE に組み込む方法を示すことにする。剛床の変換行列は、局所座標変換とは異なり、軸力と曲げモーメントが連成するため、釣合座標系を得るため全ての座標変換を変更する必要があり、多くのサブルーチンに影響する。したがって、剛床仮定の機能を組み込むために、SPACE を大幅に変更することになる。

剛床仮定を SPACE に組み込むためには、以下の部分に変更を加えることになる。

1．入力仕様
2．座標変換式の保存
3．座標変換式、特に局所座標系との関係
4．剛性行列の釣合座標系への変換
5．質量行列の釣合座標系への変換
6．右辺項の釣合座標系への変換
7．釣合座標系から部材座標系に変換
8．釣合座標系から全体座標系に変換

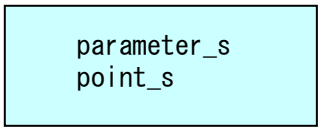
8.5.2 並列処理システムに組み込む

立体骨組の中で、床の剛性が他の部分に比較して著しく硬いとき、この床モデルでは、剛床として扱うことができる。剛床理論については、マニュアル理論編を、また、動的解析で必要となるサブルーチンの変更などについては、マニュアル動的解析編を参照されたい。ここでは、剛床の処理方法で、特に、分散並列型システム特有な処理について述べる。

ほとんどの変更部分は、動的解析と同じであり、開発環境によって、剛床処理を組み込まれたサブルーチンは、自動的に並列システムにリンクされることになる。そのため、並列用に開発されたサブルーチンあるいは、並列用に変更されたサブルーチンを更新することになる。この変更は、大きな困難を伴う作業ではない。並列処理システムで新た作成しなければならないコードは、剛床用の入力データの転送方法と、加速度

データなどのデータ転送のためのテーブル作成についてである。この 2 点について特に注意して変更する。

最初に、通常の動的解析と同様の変更を行う。まず、剛床処理を行うため、次のように 2 つの構造体の仕様拡張を行う。この構造体は、



であり、次のように変更される。

```
C     ------------------------------------------------------------
C     ●   parameter_s 構造体
C     ------------------------------------------------------------
c
c     解析パラメータ
      structure  / parameter_s/
      integer    n_unknown       !  全自由度
      integer    n_point         !  節点数
      integer    n_element       !  要素数
      integer    n_element_dll   !  DLL 用要素数
      integer    n_S_comp_model  !  任意型静的縮合モデル数
      integer    nE_New_Element  !  任意型静的縮合モデルに含まれる要素エレメント数
      integer    nM_New_Element  !  任意型静的縮合モデルに含まれる部材エレメント数
      integer    n_member        !  部材数
      integer    n_rot_axis      !  主軸回転部材数
      integer    n_local_coord   !  局所座標系を使用する節点数
      integer    n_boundary_p    !  境界節点数
      integer    nc_member       !  部材減衰機構を有する部材数
      integer    n_member_dll    !  DLL 用部材数
      integer    n_free          !  節点当たり解析自由度数
      integer    n_dim           !  解析次元数
      integer    n_skyline       !  スカイライン行列の領域数
      integer    n_sky_ave       !  平均バンド幅
      integer    n_gouyuka       !  剛床の有無
      end structure
c      record /parameter_s/ Parameter_C
c
C     ------------------------------------------------------------
C     ●   point_s 構造体
C     ------------------------------------------------------------
c
c     節点
      structure  / point_s/
      real*8   coord(3)          ! 3 次元の節点座標
      integer  irest(6)          ! 節点自由度の拘束表
      integer  local_ coord      ! 局所座標系の有無（ある場合は、回転行列の通し番号）
      integer  n_group_gouyuka   ! 剛床のグループ番号
      real*8   coord_local(3)    ! 局所座標系（全体座標系に対する角度）
      real*8   disp_initial(3)   ! 初期変位
      real*8   mass_1            ! 第一ステップ質量（ダミー）
      real*8   mass_2            ! 第二ステップ質量（ダミー）
      end structure
```

構造体要素の新規挿入

この 2 つの要素の仕様を拡張する

```
c      record /point_s/ Point
c      ALLOCATABLE ::Point(:)
c      ALLOCATE (Point(n_point))
c
```

上記２つの要素の仕様を次のように拡張する。

```
integer   local_ coord      ! 局所座標系の有無：剛床節点の有無
      剛床節点の場合は-1 をセットする。
real*8   coord_local(3)   ! 剛床節点の場合、次の座標をセットする。
        coord_local(1) : Lx
        coord_local(2) : Ly
intger  n_group_gouyuka : 剛床グループ番号
```

仕様に従って、剛床データが入力されるとき、まず、その入力データから、以下のサブルーチンを用いて解析に必要となる情報を構造体に設定する。最初に、構造データを入力するサブルーチンを以下のように変更する。

```
C      ──────────────────────────────────
C      ●   SUBROUTINE /Get_structure_pa
C        ────────────────────────────────
C      ●   構造データを入力し、バッファにデータをセットする
C      ──────────────────────────────────
       subroutine Get_structure_pa(Point,Member,Element,Parameter_C,
     *            Model_type,ierr,buf,ibuf,iis,jjs,
     *            S_comp_model,E_Fiber_work)
        .
        .
c────────────────────────────────────────────★節点データ入力
       write(damp_out,1002)
 1002  format(///1h ,'      節点座標'/)
       do i=1,node
       read(5,*,err=9912,end=9918) ii,x,y,z,idm
       write(damp_out,'(i4,3f12.3,i4)')ii,x,y,z,idm  ! ii:節点番号 idm:剛床グループ番号
       Point(ii).coord(1)  = x
       Point(ii).coord(2)  = y
       Point(ii).coord(3)  = z
       Point(ii).n_group_gouyuka = idm
       end do
c─────────────────────────★節点データバッファセット(iis=node*3)
       iis=0           !実数バッファカウント
       jjs=0           !整数バッファカウント
       do i=1,node
       jjs=jjs+1
       ibuf(jjs)=Point(i).n_group_gouyuka
       do j=1,3
```

```
        iis=iis+1
        buf(iis) = Point(i).coord(j)
        enddo
        enddo
```

上記のように、剛床のグループ番号をセットすることと、送信用バッファ領域にデータを設定する。このため、次のように、主サブルーチン submain_dynamic_a()において、送信用バッファ領域を増やす必要がある。

```
c―――――― ★★　構造データの送信用バッファ領域を確保
c      バッファデータ仕様
c      buf()     １：座標 3
c                ２：局所座標 3
c                ３：要素 17
c                ４：部材 4
c     ibuf()     １：境界拘束条件 8
c                ２：要素 6
c                ３：部材  14
c
       id_buf=Parameter_C.n_point*6+Parameter_C.n_element*17+
     *       Parameter_C.n_member*4
       jd_buf=Parameter_C.n_point*8++Parameter_C.n_element*6+
     *       Parameter_C.n_member*14
c      write(damp_out, '(3i5)') Parameter_C.n_point,
c      +                   Parameter_C.n_element, Parameter_C.n_member
       ALLOCATE (buf(id_buf+10),ibuf(jd_buf+10))
c―――――― ★★　構造データの送信用バッファ領域を確保
       call Get_structure_pa(Point,Member,Element,Parameter_C,
     *                   Model_type,ierr,buf,ibuf,id_buf,jd_buf,
     *                   S_comp_model,E_Fiber_work)
       close(nfix)
       if(ierr .ne. 0) then
       ierr_dat =iabs(ierr)
c                          err No. 12-19 使用
       call err_outf(ieer_dat)
       DEALLOCATE (buf,ibuf)
       return
       endif
c―――――――――――――――――――――――――――――――――――――――――――――――――
c
c     ★         剛床の設定
c
c―――――――――――――――――――――――――――――――――――――――――――――――――
       call Set_gouyuka(Parameter_C,Point)
```

当然、主サブルーチン内で剛床に関する情報処理を行うルーチンを加えている。

予備計算で行う Rotate_stiffness()、Rotate_damp()また、係数行列

を計算する部分で **Build_sky_mm**()、**Rotate_mass**()は、自動的に変換されている。ただし、サブルーチンコールする主サブルーチン内の引数が変わっているため、以下のように変更する。

```
c―――――――――――――――――――――――――――――――――――――――――――――――――――★剛性の釣合座標系への変換(ok)
      call Rotate_stiffness(Parameter_C,ak_linear,rot_memb,
     *                      Point,Member)
c                          write(damp_out,*) ' Rotate_stiffness Ok'
c―――――――――――――――――――――――――――――――――――――――――――――――――――★部材の減衰行列計算(ok)
      if(Parameter_C.nc_member .ne. 0) then
      call Cal_damp_linear(Element,Member,Parameter_C,ac_member,
     *           E_model6_real,work1_elememt,
     *           work2_elememt,work1_member,work2_member)
c                          write(damp_out,*) ' Cal_damp_linear Ok'
c―――――――――――――――――――――――――――――――――――――――――――――――――――★部材減衰行列の釣合座標系への変換(ok)
      call Rotate_damp(Parameter_C.n_member,ac_member,rot_memb,Member,
     *                                              Point,Parameter_C)
c                            write(damp_out,*) ' Rotate_damp Ok'
      end if
     ・
       ・
c―――――――――――――――――――――――――――――――――――――――――――――――――――★集中質量系の行列への組み込み
c―――――――――――――――――――――――――――――――――――――――――――――――――――★レーリー減衰を含む
      call Build_sky_mm(n_istep,gskym,
     *           Point,n_point, am_point , rot_local,
     *           n_local_coord ,Newmark_P, max_h_sky)
c       write(damp_out,*) ' Build_sky_mm ok'
c―――――――――――――――――――――――――――――――――――――――――――――――――――★部材の整合質量系の行列への組み込み(ok)
c―――――――――――――――――――――――――――――――――――――――――――――――――――★レーリー減衰を含む
c―――――――――――――――――――――――――――――――――――――――――――――――――――★部材の整合質量行列計算(ok*)
      if(Dynamic_load.load_mass .ne. 0) then
      call Cal_mass_linear(n_istep,Element,Member,Parameter_C,am_member,
     *           work1_elememt,work2_elememt,work1_member,work2_member,
     *           Dynamic_load.load_mass)
c       write(damp_out,*) ' Cal_mass_linear ok'
c―――――――――――――――――――――――――――――――――――――――――――――――――――★整合質量の釣合座標系への変換(ok*)
      call Rotate_mass(n_istep,Element,Member,n_member,am_member,
     *                 rot_memb,Dynamic_load.load_mass,Point)
c       write(damp_out,*) ' Rotate_mass ok'
c―――――――――――――――――――――――――――――――――――――――――――――――――――★整合質量系の組み込み(ok*)
      call Build_sky_m(n_istep,gskym,n_skyline, Member,n_member,
     *                 am_member ,Newmark_P, max_h_sky,Element)
      endif
c       write(damp_out,*) ' Build_sky_m ok'
c―――――――――――――――――――――――――――――――――――――――――――――――――――★部材減衰系の組み込み(ok)
      if(Parameter_C.nc_member .ne. 0) then
       call Build_sky_c(gskym,Member,n_member,
     *                 ac_member ,Newmark_P, max_h_sky)
c        write(damp_out,*) ' Build_sky_c ok'
      endif
c―――――――――――――――――――――――――――――――――――――――――――――――――――★線形剛性の組み込み(ok)
c―――――――――――――――――――――――――――――――――――――――――――――――――――★レーリー減衰を含む
```

```
      call Build_sky_k(n_istep,gskym,n_skyline, Member,n_member,
     *                         Ak_linear, Newmark_P, max_h_sky)
c         write(damp_out,*) ' Build_sky_k ok'
```

次は、接線剛性の座標変換について説明する。まず、該当する主サブルーチンを変更する。接線剛性を計算するサブルルーチンコールは、

```
c-------------------------------------------------★接線剛性の計算(ok)
c     write( damp_out,*)"接線剛性の計算"
c------- ★★  Time Check Codes
      call MPI_WTime(TIME_KT, nStart)                          ! 接線剛性の計算開始
c-----------------------------------------------
      call Get_nonlinear_stiff_pa(n_member1,n_member2,Control.type_analysis,
     *     Point,Parameter_C,ak_nonlinear,Member,n_member,
     *     Model_type,Element,past_disp_point,disp_point,rot_memb,
     *     E_model6_real,E_model7_real,E_model_fiber,M_model_fiber,
     *     E_model11, M_model11,
     *     E_model12, M_model12,
     *     E_model13, M_model13,
     *     E_model15, M_model15,
     *     E_model21, M_model21,
     *     E_model22, M_model22,
     *     E_model31, M_model31,
     *     E_model32, M_model32,
     *     E_model33, M_model33,
     *     MSS_work,S_comp_model,E_modelx, M_modelx,
     *     E_fiber_work, M_fiber_work,
     *     work1_elememt,work2_elememt, work1_member, work2_member )
```

であり、実際のサブルーチンは以下のようである。

```
C     ------------------------------------------------------------
C     ●   SUBROUTINE /Get_nonlinear_stiff_pa Parallel Version
C      -----------------------------------------------------------
C     ●   接線剛性行列の計算(ok)
C     ------------------------------------------------------------
      subroutine Get_nonlinear_stiff_pa(n_member1,n_member2,N_analysis,
     *     Point,Parameter_C,ak_nonlinear,Member,n_member,
     *     Model_type,Element,past_disp_point,disp_point,rot_memb,
     *     E_model6_real,E_model7_real,E_model_fiber,M_model_fiber,
     *     E_model11, M_model11,
     *     E_model12, M_model12,
     *     E_model13, M_model13,
     *     E_model15, M_model15,
     *     E_model21, M_model21,
     *     E_model22, M_model22,
     *     E_model31, M_model31,
     *     E_model32, M_model32,
     *     E_model33, M_model33,
     *     MSS_work,S_comp_model,E_modelx, M_modelx,
     *     E_fiber_work, M_fiber_work,
```

```
      *     work1_elememt,work2_elememt, work1_member, work2_member)
       .
       .
      dimension vv(12),vx(12)
      dimension Point(*),aly(2),alx(2)
       .
       .
c-------------------------------------------------★部材両端の変位取得
      do j=1,12
      ires=Member(i).irest(j)
      if(ires.gt.0) then
      v(j)=past_disp_point(ires)
      vx(j)=v(j)
      else
      v(j)=0.
      vx(j)=0.
      endif
      enddo
c-------------------------------------------------★変位を釣合系から部材座標系に変換
c-------------------------------★剛床座標変換
       do k=1,2
       i1=Member(i).nm_point(k)
       if(Point(i1).n_group_gouyuka.ne.0) then
       k1=6*(k-1)+1
       call Set_gtrans_u(1,vx(k1),Point(i1).coord_local(1),
     *                                  Point(i1).coord_local(2))
        endif
       enddo
      call RotateL_v(1,vx,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vv)
       .
       .
  100 continue
c-------------------------------------------------★部材の接線剛性を釣合系に変換

      call Rotate_K(ak,rot_memb(1,1,1,i),
     *              rot_memb(1,1,2,i),ak_nonlinear(1,1,i))
c-------------------------------------------------★剛床の変換
      if(Parameter_C.n_gouyuka .ne.0) then
      n_gouyuka=0
       do k=1,2
       i1=Member(i).nm_point(k)
       if(Point(i1).n_group_gouyuka.ne.0) then
       n_gouyuka=1
       aly(k)= Point(i1).coord_local(1)
       alx(k)= Point(i1).coord_local(2)
       else
       aly(k)=0.
       alx(k)=0.
       endif
      enddo
      if(n_gouyuka.ne.0) call Set_gtrans_k(ak_nonlinear(1,1,i),aly(1),
     *                                     alx(1), aly(2),alx(2) )
      endif
```

変位に関連して変更するサブルーチンは、以下のようであった。

```
Set_preset_disp()
Cal_stress_pa()
Check_stress_pa()
Out_disp_vel_acc()
Get_max_disp()
```

上記のサブルーチンの中で Set_preset_disp()、Out_disp_vel_acc()と Get_max_disp()は、動的解析で変更しており、自動的にリンクされる。まず、ここでは Cal_stress()について、変更部分を示すことにする。

```
      call Cal_stress_pa(n_member1,n_member2,
     *     Member,Point,n_member,Model_type,Element,
     *     past_disp_point,past_vel_point,est_ddisp_point,rot_memb,
     *     E_model6_real,ak_nonlinear)
```

```
C     ------------------------------------------------------------
C     ●   SUBROUTINE /Cal_stress_paParallel version (チェック OK)
C       ----------------------------------------------------------
C     ●   部材内応力の計算(ok)
C     ------------------------------------------------------------
      subroutine Cal_stress_pa (n_member1,n_member2,
     *                 Member,Point,n_member,Model_type,Element,
     *                 past_disp_point,past_vel_point,
     *                 est_ddisp_point,rot_memb,
     *                 E_model6_real,ak_nonlinear)
      implicit real*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
      include "..¥..¥sf3st¥submainx.h"
      record / point_s      / Point
      record / member_s     / Member
      record / element_s    / Element
      record / n_model_s    / Model_type
      record / e_model6_real_s / E_model6_real
      dimension Member(*),Element(*),E_model6_real(*)
      dimension rot_memb(3,3,2,*),ak_nonlinear(12,12,*)
      dimension Point(*)
      dimension past_disp_point(*),past_vel_point(*),est_ddisp_point(*),
     *        v(12),vv(12),vp(12),vpp(12),ak(12,12),vx(12),vpx(12)
       .
       .
CCCCCCC Parallel Code - Start CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     do i=1,n_member
      do i=n_member1,n_member2
CCCCCCC Parallel Code - End CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
c     write(76,'(a,i4)') ' mem: ',i
```

```
c————————————————————————————————★部材両端の変位取得
      do j=1,12
      ires=Member(i).irest(j)
      if(ires.gt.0) then
      vp(j)=past_disp_point(ires)
      v(j)=est_ddisp_point(ires)
      vpx(j)=vp(j)
      vx(j)=v(j)
      else
      v(j)=0.
      vp(j)=0.
      vpx(j)=0.
      vx(j)=0.
      endif
      enddo
c————————————————————————————————★変位を釣合系から部材座標系に変換
c————————————————————★剛床座標変換
       do k=1,2
       i1=Member(i).nm_point(k)
       if(Point(i1).n_group_gouyuka.ne.0) then
       k1=6*(k-1)+1
       call Set_gtrans_u(1,vx(k1),Point(i1).coord_local(1),
     *                                Point(i1).coord_local(2))
       call Set_gtrans_u(1,vpx(k1),Point(i1).coord_local(1),
     *                                Point(i1).coord_local(2))
       endif
      enddo
c————————————————————★座標変換
      call RotateL_v(1,vx,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vv)
      call RotateL_v(1,vpx,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vpp)
```

続いて、Check_stress_pa()は、以下のように変更する。

```
      call Check_stress_pa(n_member1,n_member2,Control,
     *     Control.type_analysis,Point,
     *     ak_nonlinear,Member,n_member,Model_type,
     *     Element,past_disp_point,est_ddisp_point,rot_memb,
     *     E_model6_real,E_model7_real,E_model_fiber,M_model_fiber,
     *     E_model11, M_model11,
     *     E_model12, M_model12,
     *     E_model13, M_model13,
     *     E_model15, M_model15,
     *     E_model21, M_model21,
     *     E_model22, M_model22,
     *     E_model31, M_model31,
     *     E_model32, M_model32,
     *     E_model33, M_model33,
     *     MSS_work,
     *     Bilinear_work,Trilinear_work,Concrete_work,RO_work,
     *     work1_elememt,work2_elememt, work1_member, work2_member,
     *     S_comp_model,E_modelx,M_modelx,
```

```
      *     E_fiber_work,M_fiber_work)
```

```
C     ------------------------------------------------------------------
C     ●   SUBROUTINE /Check_stress Parallel Version (チェック OK)
C       ----------------------------------------------------------------
C     ●                          部材の塑性状態をチェックする(ok)
C     ------------------------------------------------------------------
      subroutine Check_stress_pa(n_member1,n_member2,Control,N_analysis,
     *     Point,ak_nonlinear,Member,n_member,
     *     Model_type,Element,past_disp_point,disp_point,rot_memb,
     *     E_model6_real,E_model7_real,E_model_fiber,M_model_fiber,
     *     E_model11, M_model11,
     *     E_model12, M_model12,
     *     E_model13, M_model13,
     *     E_model15, M_model15,
     *     E_model21, M_model21,
     *     E_model22, M_model22,
     *     E_model31, M_model31,
     *     E_model32, M_model32,
     *     E_model33, M_model33,
     *     MSS_work,
     *     Bilinear_work,Trilinear_work,Concrete_work,RO_work,
     *     work1_elememt,work2_elememt, work1_member, work2_member ,
     *     S_comp_model,E_modelx,M_modelx,
     *     E_fiber_work,M_fiber_work)
      implicit real*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
      include "..¥..¥sf3st¥submainx.h"
      include "..¥..¥sf3st¥New_submain.h"
      record /control_s    / Control
      record / point_s     / Point
      record / member_s    / Member
      record / element_s   / Element
      record / n_model_s   / Model_type
      record / Bilinear_work_s    / Bilinear_work
      record / Trilinear_work_s   / Trilinear_work
      record / Concrete_work_s    / Concrete_work
c-------------------------------------------------★Model_No.51-70 任意要素型縮合モデル
      record / S_comp_model_s    / S_comp_model
      record / E_modelx_s        / E_modelx
      record / M_modelx_s        / M_modelx
      record / E_fiber_work_s    / E_fiber_work
      record / M_fiber_work_s    / M_fiber_work
      dimension E_modelx(*),M_modelx(*),S_comp_model(*)
      dimension E_fiber_work(*),M_fiber_work (*)
      dimension Point(*)
       .
       .
CCCCCCC Parallel Code - Start CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     do i=1,n_member
      do i=n_member1,n_member2
CCCCCCC Parallel Code - End CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
c     write(76,'(a,i4)') ' check member : ',i
```

```
c————————————————————————————————★部材両端の変位取得
      do j=1,12
      ires=Member(i).irest(j)
      if(ires.gt.0) then
      v(j)=disp_point(ires)
      vp(j)=past_disp_point(ires)
      else
      v(j)=0.
      vp(j)=0.
      endif
      enddo
c————————————————————————————————★部材両端の節点力のゼロセット
      do j=1,12
      f(j)=0.
      enddo
c————————————————————————————————★変位を釣合系から部材座標系に変換
c————————————————————————★剛床座標変換
       do k=1,2
       i1=Member(i).nm_point(k)
       if(Point(i1).n_group_gouyuka.ne.0) then
       k1=6*(k-1)+1
       call Set_gtrans_u(1,v(k1),Point(i1).coord_local(1),
     *                                  Point(i1).coord_local(2))
       call Set_gtrans_u(1,vp(k1),Point(i1).coord_local(1),
     *                                  Point(i1).coord_local(2))
       endif
       enddo
c————————————————————————————★座標変換
      call RotateL_v(1,v,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vv)
      call RotateL_v(1,vp,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vpp)
c————————————————————————————————★要素及びモデルのセット
```

次に、応力と節点力ベクトルの変換に関するサブルーチンについて検討する。最初、サブルーチン Set_point_load()は、自動的に更新されたルーチンがリンクされる。次に、右辺項ベクトルについて検討する。この部分の主ルーチンは、以下のようである。変更したサブルーチンを太字で示す。

```
c————————————————————————————————★右辺の定数ベクトルゼロセット(ok)
      call Clear_vec(n_unknown,ld_point )
c          write(damp_out,*) ' Clear_vec ok', Control.jikuzero
c————————————————————————————————★部材節点力のセット(ok)
      call Get_pointforce_ld_pa(n_member1,n_member2,ld_point,Member,
     *           Parameter_C,Point)
c     write(damp_out, *) 'Get_pointforce_ld_pa ok'
c————————————————————————————————★地震加速度セット(ok)
      acc1=Get_Acc(T,1,acc_earth,Dynamic_load,Newmark_P.f1_T)
      acc2=Get_Acc(T,2,acc_earth,Dynamic_load,Newmark_P.f1_T)
      acc3=Get_Acc(T,3,acc_earth,Dynamic_load,Newmark_P.f1_T)
c      write(damp_out,'(a,4f10.3,i4)') ' Get_Acc ok'
```

```
c------------------------------------------★集中質量に関する慣性項
      call Add_earth1_ld(n_istep,acc1,acc2,acc3,ld_point,
     *           Point,n_point,am_point,rot_local,Parameter_C)
c          write(damp_out,*) ' Add_earth1_ld ok'
c------------------------------------------★整合質量に関する慣性項
      call Add_earth2_ld(Point,acc1,acc2,acc3,ld_point,
     *    Member,n_member,am_member,rot_local,Parameter_C,Dynamic_load)
c      write(damp_out,*) ' Add_earth2_ld ok'
c-------------------------------------------------★節点荷重のセット(ok)
      p1=Get_Ps(T,1,fdd_point,Dynamic_load)
      p2=Get_Ps(T,2,fdd_point,Dynamic_load)
      p3=Get_Ps(T,3,fdd_point,Dynamic_load)
c      write(damp_out,'(a,4f10.3)') ' Get_Ps ok'
      call Add_point_ld(p1,p2,p3,ld_point,n_unknown,
     *       Dynamic_load,fld_static)
c      write(damp_out,*) ' Add_point_ld ok'
c-------------------------------------------------★線形減衰項計算(ok)
c------------------------------------------★集中質量(ok)
      call Add_damp1_ld(n_istep,ld_point,Point,n_point,
     *       past_disp_point,past_vel_point,past_acc_point,
     *       am_point,Newmark_P,Parameter_C,rot_local)
c      write(damp_out,*) ' Add_damp1_ld ok'
c------------------------------------------★整合質量(ok)
      call Add_damp2_ld_pa(n_member1,n_member2,n_istep,ld_point,Member,
     *       past_disp_point,past_vel_point,past_acc_point,
     *       am_member,Newmark_P,Element,Dynamic_load.load_mass)
c      write(damp_out,*) ' Add_damp2_ld ok'
c------------------------------------------★部材減衰(ok)
      call Add_damp3_ld_pa(n_member1,n_member2,n_istep,ld_point,Member,
     *       past_disp_point,past_vel_point,past_acc_point,
     *       ac_member,Newmark_P,Model_type.n_m_damp)
c      write(damp_out,*) ' Add_damp3_ld ok',n_istep
c-------------------------------------------------★線形剛性項計算(ok)
c                                                    レーリー減衰も含む
      call Add_stiff1_ld_pa(n_member1,n_member2,n_istep,ld_point,Member,
     *       past_disp_point,past_vel_point,past_acc_point,
     *       ak_linear,Newmark_P)
c      write(damp_out,*) ' Add_stiff1_ld ok'
c     write(damp_out, '(a)') 'Add_stiff1_ld_pa'
c     write(damp_out, '(5e16.5)') (ld_point(i), i=1, n_unknown)
c-------------------------------------------------★Δｔ秒後の変位と速度を予測(ok)
      n_err_roop = 0
 9991 continue
      nx =0
      call Estimate_disp_vel(nx, n_unknown,
     *       est_disp_point, est_vel_point, est_ddisp_point,
     *       past_disp_point, past_vel_point, past_acc_point,
     *       result_disp_point, result_vel_point,
     *       past_dacc_point,result_acc_point,Newmark_P)
c      write(damp_out,*) ' Estimate_disp_vel ok'
c------------------------------------------★解析時間のセット
c-------- ★★  Time Check Codes
      call MPI_WTime(TIME_LD, dTime_LD)
      All_Time_LD = All_Time_LD + dTime_LD!  右辺項の足し込み
```

```
c-----------------------------------------------------------------------
c
c     ★          反復計算開始
c
c-----------------------------------------------------------------------
      n_roop=Newmark_P.max_repeat
      do iroop=1,n_roop
      write(damp_out,*) ' 反復回数：' ,iroop
c-------- ★★  Time Check Codes 非線形項作成
      call MPI_WTime(TIME_NONLINER, nStart)
c-----------------------------------------------★反復に関連する右辺ベクトルのゼロセット(ok)
      call Clear_vec(n_unknown,ld_point_repeat)
c          write(damp_out,*) ' Clear_vec ok'
c-----------------------------------------------★線形剛性に関するベクトル(ok)
      call Add_stiff2_ld_pa(n_member1,n_member2,ld_point_repeat,
     *       Member,n_member,est_disp_point,ak_linear)
c      write(damp_out,*) ' Add_stiff2_ld ok'
c-----------------------------------------------★接線剛性に関する増分ベクトル(ok)
      call Add_tan_stiff_ld_pa(n_member1,n_member2,ld_point_repeat,
     *       Member,n_member,est_ddisp_point,ak_nonlinear)
c      write(damp_out,*) ' Add_tan_stiff_ld ok'
cc----------------------------------------------★Maxwell 型モデルの計算(ok)
      call Add_fdd_ld_pa(n_member1,n_member2,ld_point_repeat,
     *       E_model6_real,Element,Member,est_vel_point,rot_memb)
c      write(damp_out,*) ' Add_fdd_ld ok'
c-------- ★★  Time Check Codes
      call MPI_WTime(TIME_NONLINER, dTime_NONLINER)
      All_Time_NONLINER = All_Time_NONLINER + dTime_NONLINER  ! 非線形項の足し込み
c-----------------------------------------------★右辺項を受け取り、総和をとる
      if (n_proc .ge. 2)then
c-------- ★★スレーブより右辺項を受信
c-------- ★★  Time Check Codes
      call MPI_WTime(TIME_NET_LD, nStart)
c-------------------------------------------------
c     write(damp_out,*) ' enter recv_ld_repeat'
      call recv_ld_repeat(n_unknown,ld_point_repeat,
     +                    Ms_table, Ms_free,n_proc,
     *                    buf_disp)
c-------- ★★  Time Check Codes
                          call MPI_WTime(TIME_NET_LD, dTime_NET_LD)
      if(iroop.eq.1) then
      All_Time_NET_LD= All_Time_NET_LD + dTime_NET_LD      ! 右辺項通信
      else
      All_Time_NET_CONV=All_Time_NET_CONV+ dTime_NET_LD
      endif
c-------------------------------------------------
      endif
c     write(damp_out,*) ' recv_ld_repeat out'
c-----------------------------------------------★右辺２項の和を取る(ok)
c-------- ★★  Time Check Codes 非線形項作成
      call MPI_WTime(TIME_NONLINER, nStart)
      call add_vec(n_unknown,ld_point_repeat,ld_point)
c-------- ★★  Time Check Codes
      call MPI_WTime(TIME_NONLINER, dTime_NONLINER)
```

```
      All_Time_NONLINER = All_Time_NONLINER + dTime_NONLINER    ! 非線形項の足し込み
c-----------------------------------------------------★線形方程式を解く(ok)
c      write(76,'(a)')' ld_point_repeat'
c      write(76,'(10e12.4)')(ld_point_repeat(i),i=1,n_unknown)
c------- ★★  Time Check Codes 振動方程式を解く
      call MPI_WTime(TIME_CAL, nStart)
      n_skyline=Parameter_C.n_skyline
      call solve_sky(n_skyline,n_unknown,n_unknown,
     *             max_h_sky,gskym,gskym_d,
     *             nwork,twork,ld_point_repeat,result_acc_point)
```

```
C     ------------------------------------------------------------------
C     ●   SUBROUTINE /Get_pointforce_ld_pa
C      -----------------------------------------------------------------
C     ●   部材節点力のセット(ok)
C     ------------------------------------------------------------------
      subroutine Get_pointforce_ld_pa(n_member1,n_member2,
     *                 ld_point,Member,
     *           Parameter_C,Point)
      implicit real*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
      record / member_s   / Member
      record / point_s     / Point
      record /parameter_s  / Parameter_C
      dimension Member(*),Point(*)
      real*8 ld_point(*),ff(12)
C     ------------------------------------------------------------------
c           ld_point      :real*8 右辺項
c           Member        :structure
c           n_member      :integer 部材数
C     ------------------------------------------------------------------
c-----------------------------------------------------★座標変換なし
      if(Parameter_C.n_gouyuka .eq.0) then
      do i=n_member1,n_member2
c                        Maxwell  Model の応力は、他で考慮するのでここでは、無視する。
      if(Member(i).element_type.ne.6) then
      do j=1,12
      i1 = Member(i).irest(j)
      if(i1.gt.0) ld_point(i1)=ld_point(i1) - Member(i).force(j)
      end do
      endif
c     write(76,'(i3,12e10.3)') i,(Member(i).force(j),j=1,12)
      end do
      else
c-----------------------------------------------------★剛床座標変換あり
      do i=n_member1,n_member2
      if(Member(i).element_type.ne.6) then
      n_gouyuka=0
       do k=1,2
       i1=Member(i).nm_point(k)
       if(Point(i1).n_group_gouyuka.ne.0) n_gouyuka=1
       enddo
      if(n_gouyuka .ne. 0) then
c------------------------------------------ 剛床座標変換あり
```

```
      do j=1,12
      ff(j)=Member(i).force(j)
      enddo
       do k=1,2
       i1=Member(i).nm_point(k)
       if(Point(i1).n_group_gouyuka.ne.0) then
        k1=6*(k-1)+1
        call Set_gtrans_f(2,ff(k1),
     *     Point(i1).coord_local(1),Point(i1).coord_local(2))
        endif
      enddo
      do j=1,12
      i1 = Member(i).irest(j)
      if(i1.gt.0) ld_point(i1)=ld_point(i1) - ff(j)
      end do
c------------------------------------------ 剛床座標変換なし
      else
      do j=1,12
      i1 = Member(i).irest(j)
      if(i1.gt.0) ld_point(i1)=ld_point(i1) - Member(i).force(j)
      end do
      endif
      endif
      end do

      endif
      return
      end
```

サブルーチン Add_earth1_ld()と Add_earth2_ld()は、変更したサブルーチンが自動的にリンクされる。続いて、Add_damp1_ld()は、次のように変更される。

```
C     ------------------------------------------------------------
C     ●   SUBROUTINE /Add_damp1_ld
C     ------------------------------------------------------------
C     ●   減衰・集中質量による減衰項のセット(ok)
C     ------------------------------------------------------------
      subroutine Add_damp1_ld(nx,ld_point,Point,n_point,
     *       past_disp_point,past_vel_point,past_acc_point,
     *       am_point,Newmark_P,Parameter_C,rot_local)
      implicit real*8(A-H,O-Z)
      include "submain.h"
      record / newmark_s    / Newmark_P
      record / parameter_s /Parameter_C
      record / point_s     /Point
      dimension Point(*)
      real*8 ld_point(*),am_point(2,*),rot_local(3,3,*)
      dimension u(3),amm(3),uu(3)
      dimension past_disp_point(*),past_vel_point(*),past_acc_point(*)
c-------------------------------------------------------------------
c     nx                        :integer  第一ステップか第二ステップか
c     ld_point(*)               :real*8   右辺荷重項
```

```
c     Point                    :structure
c     n_point                  :integer  節点数
c     past_disp_point          :real*8   変位
c     past_vel_point           :real*8   速度
c     past_acc_point           :real*8   加速度
c     am_point(2,n_point)      :real*8   節点集中質量
c     Newmark_P                :structure
c     Parameter_C              :structure
c     rot_local(3,3,*)         :real*8  全体座標系から局所座標への回転行列
c-------------------------------------------------------------------------
      if(nx.eq.1) then
      a= Newmark_P.alf1_1
      b= Newmark_P.alf1_1*Newmark_P.ddt_1
      ik=1
      else
      a= Newmark_P.alf2_1
      b= Newmark_P.alf2_1*Newmark_P.ddt_1
      ik=2
      endif
c-------------------------------------------------------------------------
c-----------------------------------------------------★座標変換あり
      if(Parameter_C.n_local_coord.ne.0
     *    .or. Parameter_C.n_gouyuka .ne. 0) then
      do i=1,n_point
      am=am_point(ik,i)
      amm(1)=am
      amm(2)=am
      amm(3)=am
      if(am.ne.0.) then
      do j=1,3
      irest = Point(i).irest(j)
      if(irest.gt.0) then
      u(j)=a*past_vel_point(irest) +
     *     b*past_acc_point(irest)
      else
      u(j)=0.0
      endif
      end do
c-----------------------------------------------------★局所座標系あり
      if(Point(i).local_coord.ne.0) then
      ij=Point(i).local_coord
c-----------------------------------------------------★局所座標変換は変換前の質量剛列と同一
      do j=1,3
      u(j)=amm(j)*u(j)
      enddo
       do  j=1, 3
      i1=Point(i).irest(j)
      if(i1.gt.0) ld_point(i1)=ld_point(i1) - u(j)
      end do
c-----------------------------------------------------★剛床座標系あり
      elseif(Point(i).n_group_gouyuka.ne.0) then
      j=6
      irest = Point(i).irest(j)
      if(irest.gt.0) then
```

```
       u(j)=a*past_vel_point(irest) +
      *     b*past_acc_point(irest)
       else
       u(j)=0.0
       endif
       call Set_gtrans_u(1,u, Point(i).coord_local(1),
      *                       Point(i).coord_local(2) )
       do j=1,3
       uu(j)=amm(j)*u(j)
       enddo
       uu(6)=0.
       call Set_gtrans_f(2,uu, Point(i).coord_local(1),
      *                        Point(i).coord_local(2) )
       do  j=1, 3
       i1=Point(i).irest(j)
       if(i1.gt.0) ld_point(i1)=ld_point(i1) - uu(j)
       end do
       i1=Point(i).irest(6)
       if(i1.gt.0) ld_point(i1)=ld_point(i1) - uu(6)

c------------------------------------------------------★座標変換なし
       else
       do  j=1, 3
       i1=Point(i).irest(j)
       if(i1.gt.0) ld_point(i1)=ld_point(i1) - amm(j)*u(j)
       end do
       endif
       endif
       end do
c-----------------------------------------------------------------------
       else
       do i=1,n_point
       do j=1,3
       irest = Point(i).irest(j)
       if(irest.ne.0) then
       u(j)=a*past_vel_point(irest) +
      *     b*past_acc_point(irest)
       else
       u(j)=0.0
       endif
       end do
       am=am_point(ik,i)
       amm(1)=am
       amm(2)=am
       amm(3)=am
       do  j=1, 3
       i1=Point(i).irest(j)
       if(i1.gt.0) ld_point(i1)=ld_point(i1) - amm(j)*u(j)
       end do
       end do
c-----------------------------------------------------------------------
       endif
       return
       end
```

サブルーチン Add_damp3_ld()は、次のように変更される。

```
C     ------------------------------------------------------------------
C     ●    SUBROUTINE /Add_damp3_ld_pa (チェック OK)
C       ----------------------------------------------------------------
C     ●    部材減衰による減衰項のセット(ok)
C     ------------------------------------------------------------------
      subroutine Add_damp3_ld_pa(n_member1,n_member2,nx,ld_point,Member,
     *       past_disp_point,past_vel_point,past_acc_point,
     *       ac_member,Newmark_P,n_damp,
     *       E_model6_real,Element,Point,rot_memb)
      implicit real*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
      record / newmark_s    /Newmark_P
      record / member_s     /Member
      record / point_s      /Point
      record / element_s    / Element
      dimension Member(*),Point(*),Element(*)
      real*8 ac_nonlinear(12,12),bk(12,12)
      real*8 ld_point(*),ac_member(12,12,*)
      dimension rot_memb(3,3,2,*)
      dimension u(12),aly(2),alx(2)
      dimension E_model6_real(*)
      dimension past_disp_point(*),past_vel_point(*),past_acc_point(*)
C     ------------------------------------------------------------------
      if(n_damp.eq.0) return

      b=Newmark_P.ddt_1
      do i=n_member1,n_member2
      ij = Member(i).nm_damp
      if(ij .ne. 0) then
      do j=1,12
      irest = Member(i).irest(j)
      if(irest.gt.0) then
      u(j)=  past_vel_point(irest) +
     *     b*past_acc_point(irest)
      else
      u(j)=0.0
      endif
      end do
c      if(i.eq.239.or.i.eq.240) then
c      write(76,'(a,i4,10f12.4)') 'u(j)',i,u(1),u(7)
c      endif
c-----------------------------------------------------★Model_No.6 3 次元制震 Maxwell モデル
      if(Member(i).element_type.eq.6) then
      ien= Member(i).n_model_type
      ie = Member(i).nm_element
      ii=Element(ie).nm_type
      call Cal_nonlin_maxwelldamp(E_model6_real(ien),
     *                            bk,ii)
      call Rotate_K(bk,rot_memb(1,1,1,i),
     *              rot_memb(1,1,2,i),ac_nonlinear)
c-----------------------------------------------------★剛床変換チェック
```

```
       n_gouyuka=0
        do k=1,2
        i1=Member(i).nm_point(k)
        if(Point(i1).n_group_gouyuka.ne.0) then
        n_gouyuka=1
        aly(k)= Point(i1).coord_local(1)
        alx(k)= Point(i1).coord_local(2)
        else
        aly(k)=0.
        alx(k)=0.
        endif
       enddo
       if(n_gouyuka.ne.0) call Set_gtrans_k(ac_nonlinear,
      *                            aly(1),alx(1), aly(2),alx(2) )
c------------------------------------------------------★その他の減衰部材
       else
       do j=1,12
       do k=1,12
       ac_nonlinear(j,k)=ac_member(j,k,ij)
       enddo
       enddo
       endif
c-------------------------------------------------------
       do  j=1,12
       irest=Member(i).irest(j)
       if(irest.gt.0) then
       sum=0.
       do k=1,12
       sum=sum+ac_nonlinear(j,k)*u(k)
       enddo
       ld_point(irest)=ld_point(irest) - sum
       endif
       end do
       endif
       enddo
c-----------------------------------------------------------------------

       return
       end
```

最後に、Add_fdd_ld_pa()について変更部分を示す。

```
C     ------------------------------------------------------------
C     ●   SUBROUTINE /Add_fdd_ld_pa Parallel version (チェック OK)
C       ----------------------------------------------------------
C     ●   Maxwell 減衰項に関するベクトルを加える。(ok)
C     ------------------------------------------------------------
       subroutine Add_fdd_ld_pa(n_member1,n_member2,ld_point_repeat,
      *       E_model6_real,Element,
      *       Member,est_vel_point,rot_memb,Point)
        implicit real*8(A-H,O-Z)
        include "..¥..¥sf3st¥submain.h"
       record / member_s   / Member
```

```
      record / point_s    / Point
      record / e_model6_real_s / E_model6_real
      record /element_s / Element
      dimension Member(*),E_model6_real(*),Element(*)
      dimension Point(*)
      real*8 ld_point_repeat(*),est_vel_point(*)
      dimension rot_memb(3,3,2,*)
      dimension av(12),ud(12),vpp(12)
       .
       .
      do  j=1,12
      irest = Member(i).irest(j)
      if(irest.ne.0) then
      ud(j)=est_vel_point(irest)
      else
      ud(j)=0.
      endif
      enddo
c-------------------------------★剛床座標変換
       do k=1,2
       i1=Member(i).nm_point(k)
       if(Point(i1).n_group_gouyuka.ne.0) then
       k1=6*(k-1)+1
       call Set_gtrans_u(1,ud(k1),Point(i1).coord_local(1),
     *                                  Point(i1).coord_local(2))
       endif
      enddo
c-------------------------------★座標変換
c     write(76,'(6e12.4,3x,6e12.4)') ( ud(j),j=1,12)
      call RotateL_v(1,ud,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vpp)
c     write(76,'(6e12.4,3x,6e12.4)') ( vpp(j),j=1,12)
      if(Member(i).nm_dll_element .ne. 0) goto 9999   ! DLL 要素
      if(iett.eq.0)then
      goto(11,12,13,14,15,16,17,18,19,20),iet
   11 continue
       .
       .
  100 continue
c-----------------------------------------------------★右変更への追加
      call RotateL_v(2,av,rot_memb(1,1,1,i),rot_memb(1,1,2,i),vpp)
c-------------------------------★剛床座標変換
       do k=1,2
       i1=Member(i).nm_point(k)
       if(Point(i1).n_group_gouyuka.ne.0) then
        k1=6*(k-1)+1
        call Set_gtrans_f(2,vpp(k1),
     *     Point(i1).coord_local(1),Point(i1).coord_local(2))
       endif
      enddo
      do  j=1,12
      irest = Member(i).irest(j)
      if(irest.ne.0) then
      ld_point_repeat(irest)=ld_point_repeat(irest) - vpp(j)
      end if
```

```
      end do
      endif
      end do

      return
      end
```

8.5.3 スレーブ側への組み込み

本節では、剛床処理をスレーブ側サブルーチンに組み込む。分散並列型システムの開発環境は、動的解析サブルーチンを極力自動的に組み込むように設定されており、したがって、スレーブ側の剛床処理は、動的解析のサブルーチンを用いることで、自動的に組み込まれることになる。スレーブ側における変更部分は、スレーブ用主サブルーチンでコールしている各サブルーチンの引数を、動的解析で変更した引数に適合させることである。

スレーブ側の主サブルーチンでも、以下の剛床に関する処理を行うサブルーチンをコールする。

```
c―――――― ★★構造データ受信
      call recv_structure(buf,ibuf,id_buf,jd_buf)
c     write(*, *) ' recev_structure ok'
      call set_structure(Point,Member,Element,Parameter_C,
     *       Model_type,ierr,buf,ibuf,n_member1,n_member2,
     *       S_comp_model,E_Fiber_work)
c     write(*, *) ' set structure ok',max_member
      DEALLOCATE (buf,ibuf)

c     write(76, *) 'set_structure out::Parameter_C.n_element=',
c     *    Parameter_C.n_element
      Parameter_C.n_member = max_member     !スレーブ側で計算する担当部材数に変換
c―――――――――――――――――――――――――――――――――――――――――――――――――――――
c
c     ★         剛床の設定
c
c―――――――――――――――――――――――――――――――――――――――――――――――――――――
      call Set_gouyuka(Parameter_C,Point)
```

8.5.4 構造データの転送仕様の変更

剛床を設定するために、節点座標データの入力仕様を拡張した。分散並列処理では、このデータをマスターからスレーブに転送する必要がある。まず、マスターとスレーブの主サブルーチンの中で、バッファ領域を以下のように拡張する。

```
c———— ★★  構造データの送信用バッファ領域を確保
c     バッファデータ仕様
c     buf()     1 : 座標 3
c               2 : 局所座標 3
c               3 : 要素 17
c               4 : 部材 4
c    ibuf()     1 : 境界拘束条件 8
c               2 : 要素 6
c               3 : 部材  14
c
      id_buf=Parameter_C.n_point*6+Parameter_C.n_element*17+
     *       Parameter_C.n_member*4
      jd_buf=Parameter_C.n_point*8++Parameter_C.n_element*6+
     *       Parameter_C.n_member*14
c     write(damp_out, '(3i5)') Parameter_C.n_point,
c     +                    Parameter_C.n_element, Parameter_C.n_member
      ALLOCATE (buf(id_buf+10),ibuf(jd_buf+10))
c———— ★★  構造データの送信用バッファ領域を確保
```

次に、構造データを入力するサブルーチン Get_structure_pa()を変更する。

```
C     ——————————————————————————————————————
C     ●   SUBROUTINE /Get_structure_pa
C       ————————————————————————————————————
C     ●   構造データを入力し、バッファにデータをセットする
C     ——————————————————————————————————————
      subroutine Get_structure_pa(Point,Member,Element,Parameter_C,
     *           Model_type,ierr,buf,ibuf,iis,jjs,
     *           S_comp_model,E_Fiber_work)
       .
       .
c————————————————————————————————————————————————★節点データ入力
      write(damp_out,1002)
 1002 format(///1h ,'     節点座標'/)
      do i=1,node
      read(5,*,err=9912,end=9918) ii,x,y,z,idm
      write(damp_out,'(i4,3f12.3,i4)')ii,x,y,z,idm  ! ii:節点番号 idm:剛床グループ番号
      Point(ii).coord(1)  = x
      Point(ii).coord(2)  = y
      Point(ii).coord(3)  = z
      Point(ii).n_group_gouyuka = idm
      end do
```

スレーブ側でも同様に、set_structure()変更する。

```
C     ——————————————————————————————————————
C     ●   SUBROUTINE /set_structure
C       ————————————————————————————————————
C     ●   構造データをマスターより受信し、セットする
```

```
C     ―――――――――――――――――――――――――――――――――――――――――――――
      subroutine set_structure(Point,Member,Element,Parameter_C,
     *           Model_type,ierr,buf,ibuf,n_member1,n_member2,
     *           S_comp_model,E_Fiber_work)
       .
       .
c――――――――――――――――――――――――★節点データセット
      iis=0
      jjs=0
      do i=1,node
      jjs=jjs+1
      Point(i).n_group_gouyuka=ibuf(jjs)
      do j=1,3
      iis=iis+1
      Point(i).coord(j)=buf(iis)
      enddo
      enddo
```

これで、サブルーチン Set_gouyuka()によって、剛床に関する情報処理が行われることになる。

8.5.5 加速度データ転送のためのテーブルチェック

加速度データ転送のためのテーブル作成は、マスター側では、サブルーチン Create_table()であり、スレーブ側では Reset_ij_table()である。以下にこの 2 つのサブルーチンを示す。

```
C――――――――――――――――――――――――――――――――――――――――――――――
C     ●　SUBROUTINE /Create_table(マスター用)
C     ―――――――――――――――――――――――――
C     ●　マスター用圧縮テーブル作成
C――――――――――――――――――――――――――――――――――――――――――――――
      subroutine Create_table(Parameter_C,Member,Point,is_free,
     *      Ms_table,table_W,n_member1,n_member2)
      implicit real*8(a-h,o-z)
      include "..¥..¥sf3st¥submain.h"
      record /member_s       / Member
      record /parameter_s    / Parameter_C
      record /point_s/ Point
      dimension Member(*),Point(*)
      dimension Ms_table(*)
      integer table_W(*)

      do i=1,Parameter_C.n_point
      table_W(i)=0
      enddo
      do i=n_member1,n_member2
      do j=1,2
      is = Member(i).nm_point(j)
      table_W(is)=1
      enddo
      enddo
```

```
      is_free=0
      do i=1,Parameter_C.n_point
      if(table_W(i).ne.0) then
      do j=1,6
      if(Point(i).irest(j).gt.0) then
      is_free=is_free+1
      Ms_table(is_free)=Point(i).irest(j)
      write(76,'(a,5i12)') 'ms_table ',i,j,is_free,Ms_table(is_free)
      endif
      enddo
      endif
      enddo
      return
      end
```

```
C-----------------------------------------------------------------------
C     ●   SUBROUTINE /Reset_ij_table(スレーブ用)
C     ------------------------------------
C     ●   未知番号の取り替え
c         スレーブ側     部材番号と部材データは担当部材に変更
c                        節点番号と節点座標は変更せず
c                        釣合式の右辺項は担当部材に関連するものに圧縮
C-----------------------------------------------------------------------
      subroutine Reset_ij_table(Parameter_C,Member,Point,is_free)
      implicit real*8(a-h,o-z)
      include "..¥..¥sf3st¥submain.h"
      record /member_s       / Member
      record /parameter_s    / Parameter_C
      record /point_s/ Point
      integer, ALLOCATABLE ::table_W(:)
      dimension Member(*),Point(*)

      ALLOCATE (table_W(Parameter_C.n_point))
      n_member= Parameter_C.n_member   ! 既に担当部材数に変更済み
      do i=1,Parameter_C.n_point
      table_W(i)=0
      enddo
      do i=1,n_member
      do j=1,2
      is = Member(i).nm_point(j)
      table_W(is)=1
      enddo
      enddo
      is_free=0
      do i=1,Parameter_C.n_point
      if(table_W(i).ne.0) then
      do j=1,6
      if(Point(i).irest(j).gt.0) then
      is_free=is_free+1
      Point(i).irest(j)=is_free
      endif
      enddo
```

```
      else
      do j=1,6
      Point(i).irest(j)=0
      enddo
      endif
      enddo
C     ●　部材両端の拘束表作成(ok)
      write(76,'(//a,i4)') ' スレーブ部材両端の拘束表',n_member
      do i=1,n_member
      is = Member(i).nm_point(1)
      do k=1,6
      Member(i).irest(k)= Point(is).irest(k)
      enddo
      is = Member(i).nm_point(2)
      do k=1,6
      Member(i).irest(k+6)= Point(is).irest(k)
      enddo
      write(76,'(i4,6i6,5x,6i6)') i,(Member(i).irest(j),j=1,12)
      enddo
      DEALLOCATE (table_W)
       write(76,'(//a,i4)') ' スレーブ節点の拘束表',Parameter_C.n_point
      do i=1,Parameter_C.n_point
      write(76,'(i4,6i6,5x,6i6)') i,(Point(i).irest(j),j=1,6)
      enddo
      return
      end
```

剛床処理では、多くの節点変位同一視を行うため、このテーブルが適切に作成され、適切に使用されているかどうか確認する必要がある。ここでは、最も単純な 1 層 1 スパンの骨組みデータを用いて検討する。まず、この骨組みの節点データを示す。このデータより剛床グループが 2 つ存在することが分かる。

節点座標

1	0.000	0.000	350.000	1
2	0.000	100.000	350.000	1
3	0.000	200.000	350.000	1
4	0.000	300.000	350.000	0
5	0.000	400.000	350.000	2
6	0.000	500.000	350.000	2
7	0.000	600.000	350.000	2
8	100.000	600.000	350.000	2
9	200.000	600.000	350.000	2
10	300.000	600.000	350.000	2
11	400.000	600.000	350.000	2
12	500.000	600.000	350.000	2
13	600.000	600.000	350.000	2
14	600.000	500.000	350.000	2
15	600.000	400.000	350.000	2
16	600.000	300.000	350.000	0
17	600.000	200.000	350.000	1
18	600.000	100.000	350.000	1
19	600.000	0.000	350.000	1

20	500.000	0.000	350.000	1
21	400.000	0.000	350.000	1
22	300.000	0.000	350.000	1
23	200.000	0.000	350.000	1
24	100.000	0.000	350.000	1
25	0.000	0.000	262.500	0
26	0.000	600.000	262.500	0
27	600.000	600.000	262.500	0
28	600.000	0.000	262.500	0
29	0.000	0.000	175.000	0
30	0.000	600.000	175.000	0
31	600.000	600.000	175.000	0
32	600.000	0.000	175.000	0
33	0.000	0.000	87.500	0
34	0.000	600.000	87.500	0
35	600.000	600.000	87.500	0
36	600.000	0.000	87.500	0
37	0.000	0.000	0.000	0
38	0.000	600.000	0.000	0
39	600.000	600.000	0.000	0
40	600.000	0.000	0.000	0

次に、境界条件を示す。

境界条件

37	-1	-1	-1	-1	-1	-1
38	-1	-1	-1	-1	-1	-1
39	-1	-1	-1	-1	-1	-1
40	-1	-1	-1	-1	-1	-1

上記剛床グループの条件より、剛床の情報が作成される。

n_group_剛床数　2

n_g	1	2	-11	-12	-16	-100.00	0.00
n_g	1	3	-11	-12	-16	-200.00	0.00
n_g	1	17	-11	-12	-16	-200.00	600.00
n_g	1	18	-11	-12	-16	-100.00	600.00
n_g	1	19	-11	-12	-16	0.00	600.00
n_g	1	20	-11	-12	-16	0.00	500.00
n_g	1	21	-11	-12	-16	0.00	400.00
n_g	1	22	-11	-12	-16	0.00	300.00
n_g	1	23	-11	-12	-16	0.00	200.00
n_g	1	24	-11	-12	-16	0.00	100.00
n_g	2	6	-51	-52	-56	-100.00	0.00
n_g	2	7	-51	-52	-56	-200.00	0.00
n_g	2	8	-51	-52	-56	-200.00	100.00
n_g	2	9	-51	-52	-56	-200.00	200.00
n_g	2	10	-51	-52	-56	-200.00	300.00
n_g	2	11	-51	-52	-56	-200.00	400.00
n_g	2	12	-51	-52	-56	-200.00	500.00
n_g	2	13	-51	-52	-56	-200.00	600.00
n_g	2	14	-51	-52	-56	-100.00	600.00
n_g	2	15	-51	-52	-56	0.00	600.00

次に、かくせってん未知番号表を示す。ここでは、剛床による節点同一視が行われていることが読み取れる。

節点未知番号表　未知数：　156

1	1	2	3	4	5	6
2	1	2	7	8	9	6
3	1	2	10	11	12	6
4	13	14	15	16	17	18
5	19	20	21	22	23	24
6	19	20	25	26	27	24
7	19	20	28	29	30	24
8	19	20	31	32	33	24
9	19	20	34	35	36	24
10	19	20	37	38	39	24
11	19	20	40	41	42	24
12	19	20	43	44	45	24
13	19	20	46	47	48	24
14	19	20	49	50	51	24
15	19	20	52	53	54	24
16	55	56	57	58	59	60
17	1	2	61	62	63	6
18	1	2	64	65	66	6
19	1	2	67	68	69	6
20	1	2	70	71	72	6
21	1	2	73	74	75	6
22	1	2	76	77	78	6
23	1	2	79	80	81	6
24	1	2	82	83	84	6
25	85	86	87	88	89	90
26	91	92	93	94	95	96
27	97	98	99	100	101	102
28	103	104	105	106	107	108
29	109	110	111	112	113	114
30	115	116	117	118	119	120
31	121	122	123	124	125	126
32	127	128	129	130	131	132
33	133	134	135	136	137	138
34	139	140	141	142	143	144
35	145	146	147	148	149	150
36	151	152	153	154	155	156
37	0	0	0	0	0	0
38	0	0	0	0	0	0
39	0	0	0	0	0	0
40	0	0	0	0	0	0

マスター側で作成した加速度転送用のテーブルを以下に示す。ここで、S　未知番号はスレーブ側の未知番号をあらわし、M　未知番号はマスター側のそれを表す。ここで分かるように、太文字で示す同一視した未知番号の変位は、マスター側では同一視した変位を指すが、スレーブ側では、独立した変位として扱われており、加速度は同一視した未知番号の加速度をここでも、スレーブ側に送ることになる。したがって、スレーブ側でも同様なテーブルとなっていなければならない。逆に、右辺項は、このテーブルを使用して和をとるために、同一視の右辺項は、マスター側で同一視処理を行うことになる。

```
分担PC総数：        3 節点総数：       40

 スレーブ番号：    1  担当部材開始番号：     16  担当部材終了番号：     28
                節点番号      自由度    S__未知番号   M__未知番号
ms_table              1           1           1           1
```

ms_table	1	2	2	2
ms_table	1	3	3	3
ms_table	1	4	4	4
ms_table	1	5	5	5
ms_table	1	6	6	6
ms_table	7	1	7	19
ms_table	7	2	8	20
ms_table	7	3	9	28
ms_table	7	4	10	29
ms_table	7	5	11	30
ms_table	7	6	12	24
ms_table	13	1	13	19
ms_table	13	2	14	20
ms_table	13	3	15	46
ms_table	13	4	16	47
ms_table	13	5	17	48
ms_table	13	6	18	24
ms_table	16	1	19	55
ms_table	16	2	20	56
ms_table	16	3	21	57
ms_table	16	4	22	58
ms_table	16	5	23	59
ms_table	16	6	24	60
ms_table	17	1	**25**	**1**
ms_table	17	2	**26**	**2**
ms_table	17	3	27	61
ms_table	17	4	28	62
ms_table	17	5	29	63
ms_table	17	6	**30**	**6**
ms_table	18	1	31	1
ms_table	18	2	32	2
ms_table	18	3	33	64
ms_table	18	4	34	65
ms_table	18	5	35	66
ms_table	18	6	36	6
ms_table	19	1	37	1
ms_table	19	2	38	2
ms_table	19	3	39	67
ms_table	19	4	40	68
ms_table	19	5	41	69
ms_table	19	6	42	6
ms_table	20	1	43	1
ms_table	20	2	44	2
ms_table	20	3	45	70
ms_table	20	4	46	71
ms_table	20	5	47	72
ms_table	20	6	48	6
ms_table	21	1	49	1
ms_table	21	2	50	2
ms_table	21	3	51	73
ms_table	21	4	52	74
ms_table	21	5	53	75
ms_table	21	6	54	6
ms_table	22	1	55	1
ms_table	22	2	56	2
ms_table	22	3	57	76
ms_table	22	4	58	77
ms_table	22	5	59	78
ms_table	22	6	60	6
ms_table	23	1	61	1
ms_table	23	2	62	2
ms_table	23	3	63	79
ms_table	23	4	64	80
ms_table	23	5	65	81
ms_table	23	6	66	6

ms_table	24	1	67	1
ms_table	24	2	68	2
ms_table	24	3	69	82
ms_table	24	4	70	83
ms_table	24	5	71	84
ms_table	24	6	72	6
ms_table	25	1	73	85
ms_table	25	2	74	86
ms_table	25	3	75	87
ms_table	25	4	76	88
ms_table	25	5	77	89
ms_table	25	6	78	90
ms_table	26	1	79	91
ms_table	26	2	80	92
ms_table	26	3	81	93
ms_table	26	4	82	94
ms_table	26	5	83	95
ms_table	26	6	84	96
ms_table	27	1	85	97
ms_table	27	2	86	98
ms_table	27	3	87	99
ms_table	27	4	88	100
ms_table	27	5	89	101
ms_table	27	6	90	102
ms_table	28	1	91	103
ms_table	28	2	92	104
ms_table	28	3	93	105
ms_table	28	4	94	106
ms_table	28	5	95	107
ms_table	28	6	96	108

以下には、スレーブ側の担当部材の未知番号表と節点未知番号表を示す。

スレーブ部材両端の未知番号表　13

1	19	20	21	22	23	24	25	26	27	28	29	30
2	25	26	27	28	29	30	31	32	33	34	35	36
3	31	32	33	34	35	36	37	38	39	40	41	42
4	37	38	39	40	41	42	43	44	45	46	47	48
5	43	44	45	46	47	48	49	50	51	52	53	54
6	49	50	51	52	53	54	55	56	57	58	59	60
7	55	56	57	58	59	60	61	62	63	64	65	66
8	61	62	63	64	65	66	67	68	69	70	71	72
9	1	2	3	4	5	6	67	68	69	70	71	72
10	1	2	3	4	5	6	73	74	75	76	77	78
11	7	8	9	10	11	12	79	80	81	82	83	84
12	13	14	15	16	17	18	85	86	87	88	89	90
13	37	38	39	40	41	42	91	92	93	94	95	96

スレーブ節点の未知番号表　40

1	1	2	3	4	5	6
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	7	8	9	10	11	12
8	0	0	0	0	0	0
9	0	0	0	0	0	0
10	0	0	0	0	0	0
11	0	0	0	0	0	0
12	0	0	0	0	0	0
13	13	14	15	16	17	18

14	0	0	0	0	0	0
15	0	0	0	0	0	0
16	19	20	21	22	23	24
17	25	26	27	28	29	30
18	31	32	33	34	35	36
19	37	38	39	40	41	42
20	43	44	45	46	47	48
21	49	50	51	52	53	54
22	55	56	57	58	59	60
23	61	62	63	64	65	66
24	67	68	69	70	71	72
25	73	74	75	76	77	78
26	79	80	81	82	83	84
27	85	86	87	88	89	90
28	91	92	93	94	95	96
29	0	0	0	0	0	0
30	0	0	0	0	0	0
31	0	0	0	0	0	0
32	0	0	0	0	0	0
33	0	0	0	0	0	0
34	0	0	0	0	0	0
35	0	0	0	0	0	0
36	0	0	0	0	0	0
37	0	0	0	0	0	0
38	0	0	0	0	0	0
39	0	0	0	0	0	0
40	0	0	0	0	0	0

上記の未知番号表で分かるように、スレーブ側では、節点の同一視は行っておらず、マスター側から送られた加速度をそのまま使用することになる。ただし、既にマスター側で同一視した加速度が所定の位置に設定されている。逆に、右辺項ベクトルは、このテーブルを利用して行うため、同一視した右辺項とはならず、独立した右辺項としてマスター側に送られることになる。マスター側で同一視の処理が行われることになる。なお、右辺項の剛床変換はスレーブ側で行っている。

8.6 最大塑性率を計算するためのシステム

8.6.1 はじめに

レポーターで最大塑性率などの処理を行うために、並列処理システムにおいても、弾塑性解析を行う位置で、ひずみをファイルに出力しなければならない。しかしながら、ファイルを出力するマスター側には、全部材のひずみを保存しておらず、担当部材を有するスレーブ側に保持されている。そのため、スレーブ側のひずみをマスター側に転送するシステムを開発する必要がある。このシステムには、次に示す 2 つの問題点があり、これを解決しなければならない。

1．　単一 CPU システムでは、ひずみの保存場所は全部材に対し各部材モデルごとに確保していたが、並列処理のマスター側では、担当部材の保存領域以外の領域は存在しない。
2．　各スレーブから転送されるデータ量が一定でなく、マスターと

スレーブ両者でその大きさを計算しておかなければならない。これらの２つの問題を解決することで、部材のひずみを転送するシステムが開発可能となる。

8.6.2 コントロールデータの送受信

本節では、コントロールデータの送受信を行うサブルーチンで、変更する部分について説明する。コントロールデータを送信する個数を 2 つ増やすために、以下のように if_data の最終場所を 12 とする。

```
c————————————————————————————★解析制御情報をファイルから入力(vpp)
      ALLOCATE (ff_data(400),if_data(400),iif_dt(2))
    iif_dt(1) = 0    ! ff_data の最終場所
    iif_dt(2) = 12   ! if_data の最終場所
```

次に、コントロールデータを全スレーブに送信するサブルーチン send_ctlset()を以下に示す。変更場所を太文字で示す。

```
C     ―――――――――――――――――――――――――――――――――――――――――
C     ●   SUBROUTINE /send_ctlset
C       ―――――――――――――――――――――――――――――――――――――――
C     ●   制御情報をスレーブに転送する(ok)
C     ―――――――――――――――――――――――――――――――――――――――――
      subroutine send_ctlset(Parameter_C,fd,id,iif_dt, N_analysis,ifl)
c     外部宣言
      use MPI_DEFINE
      IMPLICIT REAL*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
c     引数
      record /parameter_s    / Parameter_C! 構造基本データ
      real*4 fd(100)
      integer :: id(100), iif_dt(2),ifl(16)
c     実装
      id(1)=N_analysis
      id(2)=Parameter_C.n_point            !node                          ! 1
      id(3)=Parameter_C.n_element          !nelem
      id(4)=Parameter_C.n_member           !memb
      id(5)=Parameter_C.n_boundary_p       !nrbound
      id(6)=Parameter_C.n_local_coord      !locod
      id(7)=Parameter_C.n_rot_axis         !njiku
      id(8)=Parameter_C.n_free             !6
c                                新規モデルのため追加
      id(9)=Parameter_C.n_S_comp_model
      id(10)=Parameter_C.nE_New_Element
      id(11)=ifl(4)                                                       ! 2
      id(12)=ifl(5)                                                       ! 3
c      write(76,'(a,2i5)') ' 制御データ',iif_dt(2),iif_dt(1)
c     write(76,'(10i8)') (id(i),i=1,iif_dt(2))
c     write(76,'(10f10.2)') (fd(i),i=1,iif_dt(1))
```

```
c―――――― ★★　制御情報と構造用制御情報を送信　　★★――――――
      call mpi_bcast(iif_dt, 2, MPI_INTEGER,ID_MASTER,                    ! 4
     +                                    MPI_COMM_WORLD,ierr)
      call mpi_bcast(id,iif_dt(2),MPI_INTEGER,ID_MASTER,                  ! 5
     +                                    MPI_COMM_WORLD,ierr)
       call mpi_bcast(fd,iif_dt(1),MPI_REAL,ID_MASTER,                    ! 6
     +                                    MPI_COMM_WORLD,ierr)
c―――――― ★★　――――――――――――――――――――――――――― ★★――――――
      return
      end
```

1. コントロールデータの整数動的領域 id()の先頭部分の 8 に、解析用のデータをセットする。同じく整数動的領域 id()の 9 番目に、新規モデルの動的領域個数をセットする。新規モデルを使用していない場合は 0 がセットされる。新規モデルのエレメント数を構造体から取り出し、その値を整数動的領域 id()の 10 番目にセットする。
2. 整数動的領域 id()の 11 番目に、部材断面ひずみの出力条件コードをセットする。
3. 整数動的領域 id()の 12 番目に、部材応力の出力条件コードをセットする
4. 次に送信する実数データ数と整数データ数を、まず、送信する。
5. 整数データを送信する。
6. 実数データを送信する。

次は、受信用サブルーチン recv_ctlset()の内容を以下に示す。

```
C     ―――――――――――――――――――――――――――――――――――――――
C     ●　SUBROUTINE /recv_ctlset
C     ―――――――――――――――――――――――――――――――――――――――
C     ●　制御ファイルをマスターより受信(ok)（スレーブ用）
C     ―――――――――――――――――――――――――――――――――――――――
      subroutine recv_ctlset(Parameter_C,ff_data,if_data,iif_dt,
     *                N_analysis,MPP_Ana_Group,ierr_dat,ifl)
c 外部宣言
      use MPI_DEFINE
      IMPLICIT REAL*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
c  引数宣言
      record /parameter_s    / Parameter_C
      integer:: if_data(*), iif_dt(2,4),ifl(16)
      real*4:: ff_data(*)
c  内部変数
      integer:: ii_dt(2)
c  本体
c―――――― ★★　制御情報と構造用制御情報を受信　　★★――――――
      call MPI_Bcast(ii_dt, 2, MPI_INTEGER,
     +               ID_MASTER, MPI_COMM_WORLD, ierr)                     ! 1
```

```
       call MPI_Bcast(if_data, ii_dt(2), MPI_INTEGER,
     +                ID_MASTER, MPI_COMM_WORLD, ierr)                    ! 2
       call MPI_Bcast(ff_data, ii_dt(1), MPI_REAL,
     +                ID_MASTER, MPI_COMM_WORLD, ierr)                    ! 3
c------- ★★ ------------------------------------ ★★-------
       N_analysis          = if_data(1)                                   ! 4
       Parameter_C.n_point = if_data(2)             !node
       Parameter_C.n_element = if_data(3)           !nelem
       Parameter_C.n_member  = if_data(4)           !memb
       Parameter_C.n_boundary_p  = if_data(5)       !nrbound
       Parameter_C.n_local_coord  = if_data(6)      !locod
       Parameter_C.n_rot_axis  = if_data(7)         !njiku
       Parameter_C.n_free  = if_data(8)             !6
c                      新規モデルのため追加
       Parameter_C.n_S_comp_model  = if_data(9)
       Parameter_C.nE_New_Element  = if_data(10)
       ifl(4)  = if_data(11)                                              ! 5
       ifl(5)  = if_data(12)                                              ! 6
       write(*,'(10i5)') (if_data(i),i=1,10)
       iif_dt(1,1)=1                                                      ! 7
       iif_dt(2,1)=13                                                     ! 8
c-------------------------------------------------★制御データの先頭番地をセット
       do i=1,3
       i1=iif_dt(1,i)
       i2=iif_dt(2,i)
       iif_dt(2,i+1)=iif_dt(2,i)+if_data(i2)+1                            ! 9
       iif_dt(1,i+1)=iif_dt(1,i)+ff_data(i1)+1                            ! 10
       enddo
       return
       end
```

1. 最初に、実際に受信する実数データ数と整数データ数を受信する。
2. 上記の値を用いて、整数データを受信する。
3. 上記の値を用いて、実数データを受信する。
4. コントロールデータの整数動的領域 id()の頭の部分から、解析用データとして該当する構造体にセットする。同じく整数動的領域 id()の 9 番目の値を、新規モデルの動的領域個数を表す構造体 Parameter_C.n_S_comp_model にセットする。整数動的領域 id()の 10 番目の値を、新規モデルのエレメント数を表す構造体 Parameter_C.nE_New_Element にセットする。
5. 部材断面ひずみの出力条件コードをセットする。
6. 部材応力の出力条件コードをセットする。
7. 最初のダイアログに関する実数データの初期値をセットする。
8. 最初のダイアログに関する整数データの**初期値 13** をセットする。
9. 第 i 番目のダイアログの整数データ個数を加える。

10. 第 i 番目のダイアログの次数データ個数を加える。

ふたつのバッファ領域には、各解析をコントロールするダイアログで定義したデータと共に、その最初の部分に当該のデータ個数がセットされている。その値を用いて上記の処理によって、制御データの先頭番地を配列 iif_dt()にセットする。

8.6.3 新たな保存領域

マスター側における全部材のひずみ保存領域を次のような動的領域を設計する。

```
c      ５） 部材
       real*8, ALLOCATABLE :: am_member(:,:,:),ac_member(:,:,:)
       real*8, save,ALLOCATABLE :: am_strain(:,:)
       integer, save,ALLOCATABLE :: nx_member(:,:)
c-------------------------------------------------------------------
c       real*8   am_member(12,12,n_member)      ! 質量整合行列（釣合座標系）
c       real*8   ac_member(12,12,nc_member)     ! 線形部材減衰行列（釣合座標系）
c       real*8   am_strain(3,*)                 ! 部材のひずみ
c       real*8   ac_member(2,nc_member)         ! 1:個数、2:ひずみ配列の番号
```

さらに、転送用のテーブルとして次の動的領域を設定する。

```
c      ０）データ取得バッファ
       integer, save, ALLOCATABLE  :: nm_parallel(:,:)
       real*8,  save, ALLOCATABLE  :: pa_work_force(:,:)
       real*4, ALLOCATABLE :: ff_data(:)
       integer, ALLOCATABLE ::if_data(:),iif_dt(:),ibuf(:)
       real*8, ALLOCATABLE :: buf(:),buf_disp(:)
       integer, ALLOCATABLE ::Ms_table(:,:),Ms_free(:)  !加速度のスレーブ転送用テーブル
       integer, save,ALLOCATABLE :: Ms_free_strain(:)  !ひずみのスレーブ転送用テーブル（転送個数）
c      １） スカイライン行列
```

上記の動的領域を確保するために、マスター側の主サブルーチンに以下のコードを挿入する。

```
c-------------------------------------------------------------------
c
c     ★          配列の大きさを動的確保する（その３）
c
c-------------------------------------------------------------------
       N=Parameter_C.n_unknown
c       ALLOCATE (test_vector(N))
       if(n_proc.ge.2)then
       ALLOCATE (Ms_table(N,n_proc-1),Ms_free(n_proc-1))
       ALLOCATE (buf_disp(N+1))
```

```
      ALLOCATE (Ms_free_strain(n_proc-1))                    ! 動的領域確保
      endif
       .
       .
c-------------------------------------------------------------------
c
c     ★         構造体の大きさを動的確保する（その１）
c
c-------------------------------------------------------------------
      ALLOCATE (Max_disp(Parameter_C.n_point))
      ALLOCATE (Member(Parameter_C.n_member))
      ALLOCATE (Max_stress(Parameter_C.n_member))
      ALLOCATE (Element(Parameter_C.n_element))
      ALLOCATE (Point(Parameter_C.n_point))
      ALLOCATE (nx_member(2,Parameter_C.n_member))            ! 動的領域確保
```

次のサブルーチンを用いて、動的領域を確保する。

```
c-----------------------------------------------★かどうかチェック(ok)
       call out_section_check_pa(Member,Element,
     *                 Parameter_C.n_member,No_section,Out_section,
     *                 ifl,iflz,i_print,n_member1,n_member2,S_comp_model
     *                 ,E_modelx,M_modelx,E_fiber_work,M_fiber_work)
c-----------------------------------------------★ファイル名一覧出力
      write(damp_out,'(//a,i5/)') ' 出力ファイル名一覧 '
     *  ,Dynamic_load.load_dynamic(1)
      do i=1,16
      write(damp_out,'(i4,3i6)') i,ifl(i),iflz(i),ifly(i)
      enddo
c-----------------------------------------------★断面ひずみの個数出力
       call Out_Strain_pa(0,Member,Element,E_model11,M_model11,
     *                 E_model12,M_model12,E_model13,M_model13,
     *                 E_model15,M_model15,
     *                 E_model21,M_model21,E_model22,M_model22,
     *                 E_model31,M_model31,E_model32,M_model32,
     *                 E_model33,M_model33,
     *                 E_model_fiber,M_model_fiber,
     *                 Parameter_C.n_member,ifl,iflz,i_print,
     *                 S_comp_model, E_modelx, M_modelx,
     *                 E_fiber_work, M_fiber_work, nx_member_strain,
     *                 am_strain,nx_member,Ms_free_strain)
c       write(damp_out,*) ' Out_Strain ok',n_member

       ALLOCATE (am_strain(3,nx_member_strain)                ! 動的領域確保
       if(n_proc.ge.2)then
       do nn_proc=1, n_proc-1
       n_member1x=nm_parallel(1, nn_proc)                     ! 担当部材の先頭番号
       n_member2x=nm_parallel(2, nn_proc)                     ! 担当部材の最終番号
       nx_proc=3*(nx_member(2, n_member2x) -
     *           nx_member(2, n_member1x) + nx_member(1, n_member2x))
       Ms_free_strain(nn_proc) = nx_proc                      ! スレーブ転送個数セット
      endif
```

8.6.4 マスター側の受信処理

サブルーチン Out_Strain_pa()では、初期設定と実際の部材ひずみをセットし、ファイルに出力する。このサブルーチンをコールする前に、スレーブからスレーブ担当部材のひずみを受け取っている必要がある。このサブルーチンは単一 CPU でのサブルーチン Out_Strain に少し変更を加えている。このサブルーチンの内容を見てみよう。

```
C     ------------------------------------------------------------------
C     ●   SUBROUTINE /Out_Strain_m_pa
C       ----------------------------------------------------------------
C     ●   部材の断面ひずみを出力(ok)
C     ------------------------------------------------------------------
      subroutine Out_Strain_m_pa(n_member1,n_member2,
     *             nx_han,Member,Element,E_model11,M_model11,
     *             E_model12,M_model12,E_model13,M_model13,
     *             E_model15,M_model15,
     *             E_model21,M_model21,E_model22,M_model22,
     *             E_model31,M_model31,E_model32,M_model32,
     *             E_model33,M_model33,
     *             E_model_fiber,M_model_fiber,
     *             n_member,ifl,iflz,i_print,
     *             S_comp_model, E_modelx, M_modelx,
     *             E_fiber_work, M_fiber_work,nx_member_strain,
     *             am_strain,nx_member,Ms_free_strain)
      implicit real*8(A-H,O-Z)
c     include "submain.h"
c     include "submainx.h"
c     include "New_submain.h"
      include "..¥..¥sf3st¥submain.h"
      include "..¥..¥sf3st¥submainx.h"
      include "..¥..¥sf3st¥New_submain.h"
c-------------------------------------------------★Model_No.51-70 任意要素型縮合モデル
      record / S_comp_model_s    / S_comp_model
      record / E_modelx_s        / E_modelx
      record / M_modelx_s        / M_modelx
      record / E_fiber_work_s    / E_fiber_work
      record / M_fiber_work_s    / M_fiber_work

      record / Member_s       / Member
      record / Element_s      / Element
      record / M_model11_s    / M_model11
      record / E_model11_s    / E_model11
      record / M_model12_s    / M_model12
      record / E_model12_s    / E_model12
      record / M_model13_s    / M_model13
      record / E_model13_s    / E_model13
      record / M_model15_s    / M_model15
      record / E_model15_s    / E_model15
      record / M_model21_s    / M_model21
      record / E_model21_s    / E_model21
      record / M_model22_s    / M_model22
      record / E_model22_s    / E_model22
```

```
      record / M_model31_s    / M_model31
      record / E_model31_s    / E_model31
      record / M_model32_s    / M_model32
      record / E_model32_s    / E_model32
      record / M_model33_s    / M_model33
      record / E_model33_s    / E_model33
      record / M_model_fiber_s   / M_model_fiber
      record / E_model_fiber_s   / E_model_fiber
      dimension E_model_fiber(*),M_model_fiber(*)
      dimension ifl(16),iflz(16)
      dimension Member(*),Element(*),M_model11(*),E_model11(*),
     *          M_model12(*),E_model12(*),M_model13(*),E_model13(*),
     *          M_model15(*),E_model15(*)
      dimension M_model21(*),E_model21(*),M_model22(*),E_model22(*)
      dimension M_model31(*),E_model31(*),M_model32(*),E_model32(*)
      dimension M_model33(*),E_model33(*)
      dimension mxtype(100),myytype(4)
      dimension E_modelx(*),M_modelx(*),S_comp_model(*)
      dimension E_fiber_work(*),M_fiber_work(*)
      real*8 am_strain(3,*)
      dimension nx_member(2,*),Ms_free_strain(*)
      data mxtype/1,1,1,1,1,1,1,1,1,1,  1,3,1,3,1,1,3,3,3,1,
     3            1,3,1,1,1,1,1,1,1,1,  1,3,1,1,1,1,1,1,1,1,
     5            1,1,1,1,1,1,1,1,1,1,  1,1,1,1,1,1,1,1,1,1,
     7            1,1,1,1,1,1,1,1,1,1,  1,1,1,1,1,1,1,1,1,1,
     9            1,1,1,1,1,1,1,1,1,1,  1,1,1,1,1,1,1,1,1,1/
      data myytype/2,4,3,5/
      real*8    v(100),vf(3)
      real*4   vf_out(3),vfx
      dimension vff(3),vfxp(3)
c     write(76,'(a,3i4)') ' out_strain',nx_han,ifl(4),n_member
c     i_print         :integer 出力制御変数 0:ファイル出力あり
c-----------------------------------------------★全部材の断面個数検索、出力
      if(ifl(4).eq.0) return
      if(nx_han.eq.0) then
      nx_member_strain=0
      do i=1,n_member
      vfx=0.
c-----------------------------------------------★断面ひずみの個数
      ie = Member(i).nm_element
      imm= Element(ie).n_element              ! 要素番号
      immm= Member(i).n_model_type              ! モデルタイプ別番号
      iet  = Member(i).element_type
      ax=(Element(ie).A*Element(ie).E)
      if(ax.ne.0.)then
      vff(1)=Element(ie).ANP/ax
      else
      vff(1)=0.
      endif
      ax=(Element(ie).Rly*Element(ie).E)
      if(ax.ne.0.)then
      vff(2)=Element(ie).AMPY/ax
      else
      vff(2)=0.
```

```
      endif
      ax=(Element(ie).Rlz*Element(ie).E)
      if(ax.ne.0.)then
      vff(3)=Element(ie).AMPZ/ax
       else
      vff(3)=0.
      endif
c      write(76,'(i4,f12.3,3(2f14.1,5x))') ie,Element(ie).E,
c     * Element(ie).A,Element(ie).ANP,
c     * Element(ie).Rly,Element(ie).AMPY,
c     + Element(ie).Rlz,Element(ie).AMPZ
       do kk=1,3
      vfxp(kk)=1./vff(kk)
      enddo
c       write(76,'(i4,4e12.4,5x,4e12.4)') i,vfx,(vff(j),j=1,3),
c     *   (vfxp(j),j=1,3)
       if(iet.eq.11) then
      vfx=2.1
      elseif(iet.eq.12) then
      vfx=3.1

       elseif(iet.eq.15) then
      vfx=2.1

      elseif(iet.eq.13) then
      vfx=2.1

      elseif(iet.eq.14) then
      vfx=3.1

       elseif(iet.eq.16) then
      vfx=2.1

      elseif(iet.eq.17) then
      vfx=1.1

      elseif(iet.eq.18.or.iet.eq.19) then
      vfx=2.1

       elseif((iet-1)/10.eq.5.or. (iet-1)/10.eq.6) then
      i_comp= iet - 50                                                    ! 1
      n_out_stress  = S_comp_model(i_comp).n_out_stress
      do m=1, n_out_stress
      kk = S_comp_model(i_comp).n_out_stress_x(m)        ! 出力エレメント番号        3
      if(kk.ne.0) then
      vfx=vfx+1.

      endif
      enddo
      vfx=vfx+0.2
      endif

      n_sec=vfx
      if(n_sec.ne.0) then
```

```
      do j=1,3
      if(vff(j).ne.0.) vff(j)=1./vff(j)
      vf_out(j)=vff(j)
       enddo
      else
      do j=1,3
      vf_out(j)=0.
      enddo
      endif
       nx_member(1,i)= n_sec
       nx_member(2,i)= nx_member_strain
       nx_member_strain = nx_member_strain + n_sec
c       write(76,'(a,2i4,3e12.4)')'nx_member',i,n_sec,(vf_out(j),j=1,3)
      write(iflz(4))vfx,(vf_out(j),j=1,3)     ! データ出力ＯＫ
      enddo
c-------------------------------------------------★断面ひずみ
      else
c       write(76,'(a,3i4)') ' i_print ',i_print,ifl(4),nx_member_strain
       if(i_print.ne.0) return
      if(ifl(4).eq.0) return
       do i= n_member1, n_member2
       nx_sec = nx_member(2,i)
       if(nx_member(1,i) .ne.0) then
c-------------------------------------------------★断面ひずみ
      ie = Member(i).nm_element
       imm= Element(ie).n_element               ! 要素番号
      immm= Member(i).n_model_type                ! モデルタイプ別番号
      iet  = Member(i).element_type
       if(iet.eq.11) then
c-------------------------------------★モデル１１
      vf(1) = M_model11(immm).d_epsi_x_1           !  軸方向歪
       vf(2) = M_model11(immm).d_epsi_y_1          !  ｙ軸に関する曲げ歪
       vf(3) = M_model11(immm).d_epsi_z_1          !  ｚ軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
       nx_sec = nx_sec+1
       do k=1,3
       am_strain(k, nx_sec)= vf(k)
       enddo
      vf(1) = M_model11(immm).d_epsi_x_2           !  軸方向歪
       vf(2) = M_model11(immm).d_epsi_y_2          !  ｙ軸に関する曲げ歪
       vf(3) = M_model11(immm).d_epsi_z_2          !  ｚ軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
       nx_sec = nx_sec+1
       do k=1,3
       am_strain(k, nx_sec)= vf(k)
       enddo
      elseif(iet.eq.12) then
c-------------------------------------★モデル１２
      vf(1) = M_model12(immm).d_epsi_x_1           !  軸方向歪
       vf(2) = M_model12(immm).d_epsi_y_1          !  ｙ軸に関する曲げ歪
       vf(3) = M_model12(immm).d_epsi_z_1          !  ｚ軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
       nx_sec = nx_sec+1
       do k=1,3
```

```
      am_strain(k, nx_sec)= vf(k)
      enddo
      vf(1) = M_model12(immm).d_epsi_x_2          !  軸方向歪
      vf(2) = M_model12(immm).d_epsi_y_2          !  y 軸に関する曲げ歪
      vf(3) = M_model12(immm).d_epsi_z_2          !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
      nx_sec = nx_sec+1
      do k=1,3
      am_strain(k, nx_sec)= vf(k)
      enddo
      vf(1) = M_model12(immm).d_epsi_x_c          !  軸方向歪
      vf(2) = M_model12(immm).d_epsi_y_c          !  y 軸に関する曲げ歪
      vf(3) = M_model12(immm).d_epsi_z_c          !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
      nx_sec = nx_sec+1
      do k=1,3
      am_strain(k, nx_sec)= vf(k)
      enddo
      elseif(iet.eq.15) then
c-------------------------------------★モデル１５

      vf(1) = M_model15(immm).d_epsi_x_1          !  軸方向歪
      vf(2) = M_model15(immm).d_epsi_y_1          !  y 軸に関する曲げ歪
      vf(3) = M_model15(immm).d_epsi_z_1          !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
      nx_sec = nx_sec+1
      do k=1,3
      am_strain(k, nx_sec)= vf(k)
      enddo
      vf(1) = M_model15(immm).d_epsi_x_2          !  軸方向歪
      vf(2) = M_model15(immm).d_epsi_y_2          !  y 軸に関する曲げ歪
      vf(3) = M_model15(immm).d_epsi_z_2          !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
        nx_sec = nx_sec+1
      do k=1,3
      am_strain(k, nx_sec)= vf(k)
      enddo
      elseif(iet.eq.13) then
c-------------------------------------★モデル２１
      vf(1) = M_model21(immm).d_epsi_x_1          !  軸方向歪
      vf(2) = M_model21(immm).d_epsi_y_1          !  y 軸に関する曲げ歪
      vf(3) = M_model21(immm).d_epsi_z_1          !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
      nx_sec = nx_sec+1
      do k=1,3
      am_strain(k, nx_sec)= vf(k)
      enddo
      vf(1) = M_model21(immm).d_epsi_x_2          !  軸方向歪
      vf(2) = M_model21(immm).d_epsi_y_2          !  y 軸に関する曲げ歪
      vf(3) = M_model21(immm).d_epsi_z_2          !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
      nx_sec = nx_sec+1
      do k=1,3
      am_strain(k, nx_sec)= vf(k)
```

```
      enddo
      elseif(iet.eq.14) then
c-------------------------------------★モデル２２
      vf(1) = M_model22(immm).d_epsi_x_1          !  軸方向歪
      vf(2) = M_model22(immm).d_epsi_y_1          !  ｙ軸に関する曲げ歪
      vf(3) = M_model22(immm).d_epsi_z_1          !  ｚ軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
      nx_sec = nx_sec+1
      do k=1,3
      am_strain(k, nx_sec)= vf(k)
      enddo
      vf(1) = M_model22(immm).d_epsi_x_2          !  軸方向歪
      vf(2) = M_model22(immm).d_epsi_y_2          !  ｙ軸に関する曲げ歪
      vf(3) = M_model22(immm).d_epsi_z_2          !  ｚ軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
      nx_sec = nx_sec+1
      do k=1,3
      am_strain(k, nx_sec)= vf(k)
      enddo
      vf(1) = M_model22(immm).d_epsi_x_c          !  軸方向歪
      vf(2) = M_model22(immm).d_epsi_y_c          !  ｙ軸に関する曲げ歪
      vf(3) = M_model22(immm).d_epsi_z_c          !  ｚ軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
      nx_sec = nx_sec+1
      do k=1,3
      am_strain(k, nx_sec)= vf(k)
      enddo
      elseif(iet.eq.16) then
c-------------------------------------★モデル３１
      vf(1) = M_model31(immm).d_epsi_x_1          !  軸方向歪
      vf(2) = M_model31(immm).d_epsi_y_1          !  ｙ軸に関する曲げ歪
      vf(3) = M_model31(immm).d_epsi_z_1          !  ｚ軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
      nx_sec = nx_sec+1
      do k=1,3
      am_strain(k, nx_sec)= vf(k)
      enddo
      vf(1) = M_model31(immm).d_epsi_x_2          !  軸方向歪
      vf(2) = M_model31(immm).d_epsi_y_2          !  ｙ軸に関する曲げ歪
      vf(3) = M_model31(immm).d_epsi_z_2          !  ｚ軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
      nx_sec = nx_sec+1
      do k=1,3
      am_strain(k, nx_sec)= vf(k)
      enddo
      elseif(iet.eq.17) then
c-------------------------------------★モデル３２
      vf(1) = M_model32(immm).d_epsi_x_1          !  軸方向歪
      vf(2) = M_model32(immm).d_epsi_y_1          !  ｙ軸に関する曲げ歪
      vf(3) = M_model32(immm).d_epsi_z_1          !  ｚ軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
      nx_sec = nx_sec+1
      do k=1,3
      am_strain(k, nx_sec)= vf(k)
```

```
      enddo
      vf(1) = M_model32(immm).d_epsi_x_2          !  軸方向歪
      vf(2) = M_model32(immm).d_epsi_y_2          !  y 軸に関する曲げ歪
      vf(3) = M_model32(immm).d_epsi_z_2          !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
      nx_sec = nx_sec+1
      do k=1,3
      am_strain(k, nx_sec)= vf(k)
      enddo
      vf(1) = M_model32(immm).d_epsi_x_c          !  軸方向歪
      vf(2) = M_model32(immm).d_epsi_y_c          !  y 軸に関する曲げ歪
      vf(3) = M_model32(immm).d_epsi_z_c          !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
      nx_sec = nx_sec+1
      do k=1,3
      am_strain(k, nx_sec)= vf(k)
      enddo
      elseif(iet.eq.18.or.iet.eq.19) then
c-------------------------------------★モデル１３
      vf(1) = M_model13(immm).d_epsi_x_1          !  軸方向歪
      vf(2) = M_model13(immm).d_epsi_y_1          !  y 軸に関する曲げ歪
      vf(3) = M_model13(immm).d_epsi_z_1          !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
      nx_sec = nx_sec+1
      do k=1,3
      am_strain(k, nx_sec)= vf(k)
      enddo
      elseif((iet-1)/10.eq.5.or. (iet-1)/10.eq.6) then
c-------------------------------------★新規モデル 51-70
      i_comp= iet - 50                                                    ! 1
      n_out_stress  = S_comp_model(i_comp).n_out_stress
      nmmx= Member(i).n_model_type -1              ! M_Fiber_work の開始番号   2
      nmx = Element(ie).n_section(1) -1            ! E_Fiber_work の開始番号
      do m=1, n_out_stress
      kk = S_comp_model(i_comp).n_out_stress_x(m)    ! 出力エレメント番号      3
      if(kk.ne.0) then
      immmx= M_Fiber_work(nmmx+kk).nm_section        ! 内部エレメント番号
      vf(1) = M_modelx(immmx).d_epsi_x               !  軸方向歪
      vf(2) = M_modelx(immmx).d_epsi_y               !  y 軸に関する曲げ歪
      vf(3) = M_modelx(immmx).d_epsi_z               !  z 軸に関する曲げ歪
c      write(iflz(4))(vf(k),k=1,3)                   ! ひずみ              ! 9
      nx_sec = nx_sec+1
      do k=1,3
      am_strain(k, nx_sec)= vf(k)
      enddo
      endif
      enddo
      endif
c-------------------------------------★断面ひずみ出力終わり
c     write(76,'(a,2i4,3e14.5)') ' st ',i,nx_sec,(vf(k),k=1,3)
      endif

      enddo
c-------------------------------------★データの出力
```

```
c      write(76,'(a,i4,3e14.5)')' strain ',nx_member_strain
       do i=1, nx_member_strain
       do k=1,3
       vf_out(k)= am_strain(k, i)
       enddo

c      write(76,'(a,i4,3e14.5)')' strain ',i,( vf_out(k),k=1,3)
       write(iflz(4))( vf_out(k),k=1,3)
       enddo
      endif
      return
     end
```

次に、スレーブからひずみを受け取るサブルーチンを設計する。ここでは、各スレーブから送られてくる個数が異なることに注意する。

```
C-------------------------------------------------------------------
C     ●   SUBROUTINE / recv_strain_vpp
C     ------------------------
C     ●  各スレーブが計算したひずみをスレーブから受け取る
C-------------------------------------------------------------------
      subroutine recv_strain_vpp(nx_member, Ms_free_strain,
     +               am_strain, pa_work_strain ,nm_parallel, n_proc)
c  外部宣言
      use MPI_DEFINE
      implicit none
      include "..¥..¥sf3st¥submain.h"
c      引数宣言
c      pa_work_stress       : real*4   ひずみの転送用領域
c      n_member1            : integer  担当部材の先頭番号
c      n_member2            : integer  担当部材の最終番号
c      n_id                 : integer  プロセスナンバー
      rial*4 am_strain(3,*)
      dimension nx_member(2,*),Ms_free_strain(*)
      real*8:: pa_work_strain(3,*)
      integer :: n_proc
      integer :: nm_parallel(2, 0:n_proc)
c      内部変数宣言
      integer::n_member1,n_member2
      integer, save :: n_Maxcount
      integer :: recv_status(MPI_STATUS_SIZE)! 受信ステータスを宣言
      integer:: k,i,j
      integer:: ii
      integer:: ierr
c      実装
       do k=1,n_proc-1
       n_member1 = nm_parallel(1,k  )
       n_member2 = nm_parallel(2,k  )
       n_Maxcount= Ms_free_strain(k)!  担当部材のひずみ個数
       call MPI_Recv(pa_work_force,n_Maxcount MPI_DOUBLE,
     +              k, MPI_ANY_TAG, MPI_COMM_WORLD, recv_status,ierr)
c      write(76,'(/a,10i4)') ' proc ',k,n_Maxcount,n_member1,
c     *   n_member2
```

```
      j=0
      do i=n_member1,n_member2
      if(nx_member(1,i).ne.0)then
      nx_sec= nx_member(2,i)
      do kk=1, nx_member(1,i)
      nx_sec = nx_sec+1
      j=j+1
      do m=1,3
      am_strain(m, nx_sec)=pa_work_strain(m,j)
      enddo
      enddo
      enddo
      enddo
      return
      end
```

上記 2 つのサブルーチンを次のようにマスター側の主サブルーチンでコールする。

```
c-------------------------------------------------------★部材応力を出力
      call Out_Fiber(Member,Element,E_model11,M_model11,
     *               E_model12,M_model12,E_model13,M_model13,
     *               E_model15,M_model15,
     *               E_model21,M_model21,E_model22,M_model22,
     *               E_model31,M_model31,E_model32,M_model32,
     *               E_model33,M_model33,
     *               E_model_fiber,M_model_fiber,
     *               n_member,ifl,iflz,i_print,Out_section,
     *               S_comp_model, E_modelx, M_modelx,
     *               E_fiber_work, M_fiber_work)
c       write(damp_out,*) ' Out_stress ok'
c-------------------------------------------------------★部材部材ひずみをスレーブより受信
      call recv_strain_vpp(nx_member, Ms_free_strain,
     +                am_strain, pa_work_strain ,nm_parallel, n_proc)
c       write(damp_out,*) ' recv_force_ strain ok'
c-------------------------------------------------------★部材ひずみを出力
      call Out_Strain_m_pa(1,Member,Element,E_model11,M_model11,
     *               E_model12,M_model12,E_model13,M_model13,
     *               E_model15,M_model15,
     *               E_model21,M_model21,E_model22,M_model22,
     *               E_model31,M_model31,E_model32,M_model32,
     *               E_model33,M_model33,
     *               E_model_fiber,M_model_fiber,
     *               Parameter_C.n_member,ifl,iflz,i_print,
     *               S_comp_model, E_modelx, M_modelx,
     *               E_fiber_work, M_fiber_work, nx_member_strain,
     *               am_strain,nx_member,Ms_free_strain)
c       write(damp_out,*) ' Out_Strain ok' ,n_member
```

8.6.5 スレーブ側の送信処理

スレーブ側では、マスター用のサブルーチンを少し変更することで対

処し得る。変更したサブルーチンを以下の示そう。

```
C     ------------------------------------------------------------------
C     ●    SUBROUTINE /Out_Strain_pa
C       ----------------------------------------------------------------
C     ●    部材の断面ひずみを出力(ok)
C     ------------------------------------------------------------------
      subroutine Out_Strain_pa(nx_han,Member,
     *             Element,E_model11,M_model11,
     *             E_model12,M_model12,E_model13,M_model13,
     *             E_model15,M_model15,
     *             E_model21,M_model21,E_model22,M_model22,
     *             E_model31,M_model31,E_model32,M_model32,
     *             E_model33,M_model33,
     *             E_model_fiber,M_model_fiber,
     *             n_member,ifl,iflz,i_print,
     *             S_comp_model, E_modelx, M_modelx,
     *             E_fiber_work, M_fiber_work, nx_member_strain,
     *             am_strain)
      implicit real*8(A-H,O-Z)
      include "..¥..¥sf3st¥submain.h"
      include "..¥..¥sf3st¥submainx.h"
      include "..¥..¥sf3st¥New_submain.h"
c-----------------------------------------------★Model_No.51-70 任意要素型縮合モデル
      record / S_comp_model_s    / S_comp_model
      record / E_modelx_s        / E_modelx
      record / M_modelx_s        / M_modelx
      record / E_fiber_work_s    / E_fiber_work
      record / M_fiber_work_s    / M_fiber_work

      record / Member_s       / Member
      record / Element_s      / Element
      record / M_model11_s    / M_model11
      record / E_model11_s    / E_model11
      record / M_model12_s    / M_model12
      record / E_model12_s    / E_model12
      record / M_model13_s    / M_model13
      record / E_model13_s    / E_model13
      record / M_model15_s    / M_model15
      record / E_model15_s    / E_model15
      record / M_model21_s    / M_model21
      record / E_model21_s    / E_model21
      record / M_model22_s    / M_model22
      record / E_model22_s    / E_model22
      record / M_model31_s    / M_model31
      record / E_model31_s    / E_model31
      record / M_model32_s    / M_model32
      record / E_model32_s    / E_model32
      record / M_model33_s    / M_model33
      record / E_model33_s    / E_model33
      record / M_model_fiber_s   / M_model_fiber
      record / E_model_fiber_s   / E_model_fiber
      dimension E_model_fiber(*),M_model_fiber(*)
```

```
      dimension ifl(16),iflz(16)
      dimension Member(*),Element(*),M_model11(*),E_model11(*),
     *          M_model12(*),E_model12(*),M_model13(*),E_model13(*),
     *          M_model15(*),E_model15(*)
       dimension M_model21(*),E_model21(*),M_model22(*),E_model22(*)
       dimension M_model31(*),E_model31(*),M_model32(*),E_model32(*)
       dimension M_model33(*),E_model33(*)
       dimension mxtype(100),myytype(4)
       dimension E_modelx(*),M_modelx(*),S_comp_model(*)
       dimension E_fiber_work(*),M_fiber_work(*)
       real*8 am_strain(3,*)
      data mxtype/1,1,1,1,1,1,1,1,1,1,  1,3,1,3,1,1,3,3,3,1,
     3            1,3,1,1,1,1,1,1,1,1,  1,3,1,1,1,1,1,1,1,1,
     5            1,1,1,1,1,1,1,1,1,1,  1,1,1,1,1,1,1,1,1,1,
     7            1,1,1,1,1,1,1,1,1,1,  1,1,1,1,1,1,1,1,1,1,
     9            1,1,1,1,1,1,1,1,1,1,  1,1,1,1,1,1,1,1,1,1/
       data myytype/2,4,3,5/
      real*8    v(100),vf(3),vfx
      dimension vff(3),vfxp(3)
c     write(76,'(a,3i4)') ' out_strain',nx_han,ifl(4),n_member
c      i_print         :integer 出力制御変数 0:ファイル出力あり
c-------------------------------------------------★全部材の断面個数検索、出力
       if(nx_han.eq.0.and.ifl(4).ne.0) then
       nx_member_strain=0
       do i=1,n_member
      n_sec=0
c-------------------------------------------------★断面ひずみの個数
      iet  = Member(i).element_type
       if(iet.eq.11) then
      n_sec=2
      elseif(iet.eq.12) then
      n_sec=3
       elseif(iet.eq.15) then
      n_sec=2
      elseif(iet.eq.13) then
      n_sec=2
      elseif(iet.eq.14) then
      n_sec=3
       elseif(iet.eq.16) then
      n_sec=2
      elseif(iet.eq.17) then
      n_sec=1
      elseif(iet.eq.18.or.iet.eq.19) then
      n_sec=2
       elseif((iet-1)/10.eq.5.or. (iet-1)/10.eq.6) then
      i_comp= iet - 50                                                    ! 1
       n_out_stress  = S_comp_model(i_comp).n_out_stress
       do m=1, n_out_stress
       kk = S_comp_model(i_comp).n_out_stress_x(m)        ! 出力エレメント番号        3
       if(kk.ne.0) then
       n_sec=n_sec+1
       endif
       enddo
       endif
```

```
       nx_member_strain = nx_member_strain + n_sec
c      write(76,'(a,3i4)') 'nx_member_strain',nx_member_strain,n_sec,iet
       enddo

c-------------------------------------------------★断面ひずみ
       else
        if(i_print.ne.0) return
       if(ifl(4).eq.0) return
        nx_sec=0
        do i= 1, n_member
c-------------------------------------------------★断面ひずみ
       ie = Member(i).nm_element
        imm= Element(ie).n_element                ! 要素番号
       immm= Member(i).n_model_type                 ! モデルタイプ別番号
       iet  = Member(i).element_type
        if(iet.eq.11) then
c-----------------------------------★モデル１１
       vf(1) = M_model11(immm).d_epsi_x_1            !  軸方向歪
        vf(2) = M_model11(immm).d_epsi_y_1            !  y軸に関する曲げ歪
        vf(3) = M_model11(immm).d_epsi_z_1            !  z軸に関する曲げ歪
c      write(iflz(4))(vf(k),k=1,3)
        nx_sec = nx_sec+1
        do k=1,3
        am_strain(k, nx_sec)= vf(k)
        enddo
c       write(76,'(a,2i4,3e13.4)') ' st_int ',i,nx_sec,(vf(k),k=1,3)
       vf(1) = M_model11(immm).d_epsi_x_2            !  軸方向歪
        vf(2) = M_model11(immm).d_epsi_y_2            !  y軸に関する曲げ歪
        vf(3) = M_model11(immm).d_epsi_z_2            !  z軸に関する曲げ歪
c      write(iflz(4))(vf(k),k=1,3)
        nx_sec = nx_sec+1
        do k=1,3
        am_strain(k, nx_sec)= vf(k)
        enddo
c       write(76,'(a,2i4,3e13.4)') ' st_int ',i,nx_sec,(vf(k),k=1,3)
       elseif(iet.eq.12) then
c-----------------------------------★モデル１２
       vf(1) = M_model12(immm).d_epsi_x_1            !  軸方向歪
        vf(2) = M_model12(immm).d_epsi_y_1            !  y軸に関する曲げ歪
        vf(3) = M_model12(immm).d_epsi_z_1            !  z軸に関する曲げ歪
c      write(iflz(4))(vf(k),k=1,3)
        nx_sec = nx_sec+1
        do k=1,3
        am_strain(k, nx_sec)= vf(k)
        enddo
       vf(1) = M_model12(immm).d_epsi_x_2            !  軸方向歪
        vf(2) = M_model12(immm).d_epsi_y_2            !  y軸に関する曲げ歪
        vf(3) = M_model12(immm).d_epsi_z_2            !  z軸に関する曲げ歪
c      write(iflz(4))(vf(k),k=1,3)
        nx_sec = nx_sec+1
        do k=1,3
        am_strain(k, nx_sec)= vf(k)
        enddo
```

```
      vf(1) = M_model12(immm).d_epsi_x_c            !  軸方向歪
       vf(2) = M_model12(immm).d_epsi_y_c           !  y 軸に関する曲げ歪
       vf(3) = M_model12(immm).d_epsi_z_c           !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
       nx_sec = nx_sec+1
       do k=1,3
       am_strain(k, nx_sec)= vf(k)
       enddo
       elseif(iet.eq.15) then
c-------------------------------------★モデル１５

      vf(1) = M_model15(immm).d_epsi_x_1            !  軸方向歪
       vf(2) = M_model15(immm).d_epsi_y_1           !  y 軸に関する曲げ歪
       vf(3) = M_model15(immm).d_epsi_z_1           !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
       nx_sec = nx_sec+1
       do k=1,3
       am_strain(k, nx_sec)= vf(k)
       enddo
      vf(1) = M_model15(immm).d_epsi_x_2            !  軸方向歪
       vf(2) = M_model15(immm).d_epsi_y_2           !  y 軸に関する曲げ歪
       vf(3) = M_model15(immm).d_epsi_z_2           !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
         nx_sec = nx_sec+1
       do k=1,3
       am_strain(k, nx_sec)= vf(k)
       enddo
       elseif(iet.eq.13) then
c-------------------------------------★モデル２１
      vf(1) = M_model21(immm).d_epsi_x_1            !  軸方向歪
       vf(2) = M_model21(immm).d_epsi_y_1           !  y 軸に関する曲げ歪
       vf(3) = M_model21(immm).d_epsi_z_1           !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
       nx_sec = nx_sec+1
       do k=1,3
       am_strain(k, nx_sec)= vf(k)
       enddo
      vf(1) = M_model21(immm).d_epsi_x_2            !  軸方向歪
       vf(2) = M_model21(immm).d_epsi_y_2           !  y 軸に関する曲げ歪
       vf(3) = M_model21(immm).d_epsi_z_2           !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
       nx_sec = nx_sec+1
       do k=1,3
       am_strain(k, nx_sec)= vf(k)
       enddo
      elseif(iet.eq.14) then
c-------------------------------------★モデル２２
      vf(1) = M_model22(immm).d_epsi_x_1            !  軸方向歪
       vf(2) = M_model22(immm).d_epsi_y_1           !  y 軸に関する曲げ歪
       vf(3) = M_model22(immm).d_epsi_z_1           !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
       nx_sec = nx_sec+1
       do k=1,3
       am_strain(k, nx_sec)= vf(k)
```

```
       enddo
      vf(1) = M_model22(immm).d_epsi_x_2          !  軸方向歪
       vf(2) = M_model22(immm).d_epsi_y_2          !  y 軸に関する曲げ歪
       vf(3) = M_model22(immm).d_epsi_z_2          !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
       nx_sec = nx_sec+1
       do k=1,3
       am_strain(k, nx_sec)= vf(k)
       enddo
      vf(1) = M_model22(immm).d_epsi_x_c          !  軸方向歪
       vf(2) = M_model22(immm).d_epsi_y_c          !  y 軸に関する曲げ歪
       vf(3) = M_model22(immm).d_epsi_z_c          !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
       nx_sec = nx_sec+1
       do k=1,3
       am_strain(k, nx_sec)= vf(k)
       enddo
       elseif(iet.eq.16) then
c-------------------------------------★モデル３１
      vf(1) = M_model31(immm).d_epsi_x_1          !  軸方向歪
       vf(2) = M_model31(immm).d_epsi_y_1          !  y 軸に関する曲げ歪
       vf(3) = M_model31(immm).d_epsi_z_1          !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
       nx_sec = nx_sec+1
       do k=1,3
       am_strain(k, nx_sec)= vf(k)
       enddo
      vf(1) = M_model31(immm).d_epsi_x_2          !  軸方向歪
       vf(2) = M_model31(immm).d_epsi_y_2          !  y 軸に関する曲げ歪
       vf(3) = M_model31(immm).d_epsi_z_2          !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
       nx_sec = nx_sec+1
       do k=1,3
       am_strain(k, nx_sec)= vf(k)
       enddo
      elseif(iet.eq.17) then
c-------------------------------------★モデル３２
      vf(1) = M_model32(immm).d_epsi_x_1          !  軸方向歪
       vf(2) = M_model32(immm).d_epsi_y_1          !  y 軸に関する曲げ歪
       vf(3) = M_model32(immm).d_epsi_z_1          !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
       nx_sec = nx_sec+1
       do k=1,3
       am_strain(k, nx_sec)= vf(k)
       enddo
      vf(1) = M_model32(immm).d_epsi_x_2          !  軸方向歪
       vf(2) = M_model32(immm).d_epsi_y_2          !  y 軸に関する曲げ歪
       vf(3) = M_model32(immm).d_epsi_z_2          !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
       nx_sec = nx_sec+1
       do k=1,3
       am_strain(k, nx_sec)= vf(k)
       enddo
      vf(1) = M_model32(immm).d_epsi_x_c          !  軸方向歪
```

```
       vf(2) = M_model32(immm).d_epsi_y_c            !  y 軸に関する曲げ歪
       vf(3) = M_model32(immm).d_epsi_z_c            !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
       nx_sec = nx_sec+1
       do k=1,3
       am_strain(k, nx_sec)= vf(k)
       enddo
      elseif(iet.eq.18.or.iet.eq.19) then
c-------------------------------------★モデル１３
      vf(1) = M_model13(immm).d_epsi_x_1            !  軸方向歪
       vf(2) = M_model13(immm).d_epsi_y_1            !  y 軸に関する曲げ歪
       vf(3) = M_model13(immm).d_epsi_z_1            !  z 軸に関する曲げ歪
c     write(iflz(4))(vf(k),k=1,3)
       nx_sec = nx_sec+1
       do k=1,3
       am_strain(k, nx_sec)= vf(k)
       enddo
       elseif((iet-1)/10.eq.5.or. (iet-1)/10.eq.6) then
c-------------------------------------★新規モデル 51-70
       i_comp= iet - 50                                                    ! 1
       n_out_stress  = S_comp_model(i_comp).n_out_stress
       nmmx= Member(i).n_model_type -1                  ! M_Fiber_work の開始番号   2
       nmx = Element(ie).n_section(1) -1                ! E_Fiber_work の開始番号
       do m=1, n_out_stress
       kk = S_comp_model(i_comp).n_out_stress_x(m)       ! 出力エレメント番号        3
       if(kk.ne.0) then
       immmx= M_Fiber_work(nmmx+kk).nm_section           ! 内部エレメント番号
       vf(1) = M_modelx(immmx).d_epsi_x                  !  軸方向歪
       vf(2) = M_modelx(immmx).d_epsi_y                  !  y 軸に関する曲げ歪
       vf(3) = M_modelx(immmx).d_epsi_z                  !  z 軸に関する曲げ歪
c      write(iflz(4))(vf(k),k=1,3)                       ! ひずみ               ! 9
       nx_sec = nx_sec+1
       do k=1,3
       am_strain(k, nx_sec)= vf(k)
       enddo
       endif
       enddo
       endif
c-----------------------------------------------★断面ひずみ出力終わり
       enddo
c      write(76,'(a,3i4)') ' nx_sec ',nx_sec
      endif
      return
      end
```

上記サブルーチンを、スレーブ側の主サブルーチンで以下のようにコールする。最初は、初期設定と送信すべきデータ量を設定する。

```
C-----------------------------------------------★出力指定した断面がファイバー要素
c-----------------------------------------------★断面ひずみの個数出力
      call Out_Strain_pa(0,Member,Element,E_model11,M_model11,
     *                   E_model12,M_model12,E_model13,M_model13,
     *                   E_model15,M_model15,
```

```
     *                E_model21,M_model21,E_model22,M_model22,
     *                E_model31,M_model31,E_model32,M_model32,
     *                E_model33,M_model33,
     *                E_model_fiber,M_model_fiber,
     *                Parameter_C.n_member,ifl,iflz,i_print,
     *                S_comp_model, E_modelx, M_modelx,
     *                E_fiber_work, M_fiber_work, nx_member_strain,
     *                pa_work_force)
```

上のサブルーチンで分かるように、送信するデータのワーク領域として、部材の応力を送信するワーク領域である pa_work_force を利用している。一般には、ひずみが必要とする領域よりも、応力のそれのほうが大きい。しかし、任意型の部材モデルでは、同様な状態となるか不定であるので、先のサブルーチンの後で、次のコードで、その大きさを変更することにした。

```
      if(nx_member_strain*3 .gt. 33*max_member) then
      DEALLOCATE (pa_work_force)
     ALLOCATE (pa_work_force(3,nx_member_strain))
      endif
```

また、スレーブ側で、計算したひずみをワーク領域にセットした後、マスター側に送信するサブルーチンコールを示す。

```
c-------------------------------------------------★部材の節点力と部材応力をマスターに送る(vpp)
c------- ★★マスターに部材応力を送信（マスターがファイル出力する時のみ）
       if(i_print.eq.0) then
      call send_force_vpp(n_member,Member,pa_work_force,ifl(5))
c-------------------------------------------------★部材ひずみを出力
      call Out_Strain_pa(1,Member,Element,E_model11,M_model11,
     *                E_model12,M_model12,E_model13,M_model13,
     *                E_model15,M_model15,
     *                E_model21,M_model21,E_model22,M_model22,
     *                E_model31,M_model31,E_model32,M_model32,
     *                E_model33,M_model33,
     *                E_model_fiber,M_model_fiber,
     *                Parameter_C.n_member,ifl,iflz,i_print,
     *                S_comp_model, E_modelx, M_modelx,
     *                E_fiber_work, M_fiber_work, nx_member_strain,
     *                pa_work_force)
c       write(damp_out,*) ' Out_Strain ok',n_member
c-------------------------------------------------★部材部材ひずみをマスターに送信
      call send_strain_vpp(nx_member_strain,
     *                           pa_work_force,ifl(4))
```

最後に、得られたひずみをマスター側に送信するサブルーチンを示すことにする。

```
C-----------------------------------------------------------------------
C     ●   SUBROUTINE /send_strain_vpp（スレーブ用）
C     ―――――――――――――――――――――
C     ●  各スレーブが計算したひずみをマスターに送る
C-----------------------------------------------------------------------
      subroutine send_strain_vpp(nx_member_strain,
     *               pa_work_force,ifl)
c     外部宣言
      use MPI_DEFINE
      implicit none
      include "..¥..¥sf3st¥submain.h"
c     引数宣言
      real*8:: pa_work_force(3,*)
      integer:: ifl
c     内部変数
      integer:: n_count,nx_member_strain   ! 配列の数
      integer:: i,j
      integer:: ierr
c     本体
       if(ifl.eq.0) return
      n_count= nx_member_strain*3
c        write(76,'(a,i4)') ' send strain ',n_count
      call MPI_Send(pa_work_force,n_count,MPI_DOUBLE,
     +            ID_MASTER, MPI_ANY_TAG, MPI_COMM_WORLD, ierr)
      return
      end
```